Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система для організації практичних занять в автошколі. Back-end.
(тема)

Виконав:
студент 4 курсу, групи ПЗПІ-20-5

Охотникова О.С.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник ст.викл. кафедри ПІ Олійник О.В.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Охотниковій Олександрі Сергіївні
(прізвище, ім’я, по батькові)

1. Тема роботи Програмна система для організації практичних занять в автошколі. Back-end

Затверджена наказом по університету від 20.05. 2024р. № 471 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 20.06.2024

3. Вихідні дані до роботи Розробити серверну частину комплексної програмної системи, яка призначена для організації практичних занять в автошколі. Використовувати мову програмування JavaScript та платформу Node.js.

4. Перелік питань, що потрібно опрацювати в роботі

Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, перелік джерел посилань, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	21.05.2024	*виконано*
2	Створення специфікації ПЗ	22.05.2024	*виконано*
3	Проектування ПЗ	24.05.2024	*виконано*
4	Розробка ПЗ	08.05.2024	*виконано*
5	Тестування ПЗ	09.05.2024	*виконано*
6	Оформлення пояснювальної записки	11.06.2024	*виконано*
7	Підготовка презентації та доповіді	13.06.2024	*виконано*
8	Попередній захист	14.06.2024	*виконано*
9	Нормоконтроль, рецензування	15.06.2024	*виконано*
10	Здача роботи у електронний архів	16.06.2024	*виконано*
11	Допуск до захисту у зав. кафедри	19.06.2024	*виконано*

Дата видачі завдання 21 травня 2024р.

Студент (ка)_____ Охотникова О.С.

(підпис)

Керівник роботи _____ ст.викл. кафедри ПІ Олійник О.В.

(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра містить: 65 с., 19 рис., 10 джерел.

АВТОМОБІЛЬ, АВТОШКОЛА, АДМІНІСТРАТОР, БАЗА ДАНИХ, ВІДГУКИ, ІНСТРУКТОР, ПРАКТИЧНЕ ЗАНЯТТЯ, СЕРВЕРНА ЧАСТИНА, СТУДЕНТ, EXPRESS, JAVASCRIPT, MONGODB, NODE.JS, REST API.

Об’єкт розробки – серверна частина програмної системи для організації практичних занять в автошколі, яка дозволяє студентам автошколи обрати інструктора та переглядати його розклад, дає змогу записатись на заняття з водіння у обрані день та час, а також полегшує комунікацію з інструктором.

Об’єктом дослідження є предметна область з практичних занять з водіння в автошколі, досягнення зручного способу вибору, запису та відстеження занять та розкладу для студентів та інструкторів.

Методи рішення завдання включають аналіз та моделювання предметної області, концептуальне моделювання, UML-Моделювання, використання MongoDB Atlas, розробка серверної частини програмної системи на основі REST архітектури за допомогою Express фреймворку.

Результатом роботи є програмна система для зручної організації практичних занять в автошколі як для студентів, так і для інструкторів.

CAR, DRIVING SCHOOL, ADMINISTRATOR, DATABASE, FEEDBACK, INSTRUCTOR, PRACTICAL LESSON, SERVER PART, STUDENT, EXPRESS, JAVASCRIPT, MONGODB, NODE.JS, REST API.

The object of development is the server side of the software system for organizing practical lessons in a driving school, which allows driving school students to choose an instructor and view his schedule, allows students to sign up for driving

lessons on the chosen day and time, and also facilitates communication with the instructor.

The object of study is the subject area of practical driving lessons in a driving school, achieving a convenient way of choosing, signing up and tracking lessons and schedules for students and instructors.

Task solution methods include analysis and modeling of the subject area, conceptual modeling, UML-Modeling, use of MongoDB Atlas, development of the server part of the software system based on REST architecture using the Express framework.

The result of the work is a software system for the convenient organization of practical lessons in a driving school for both students and instructors.

Я, Охотникова Олександра Сергіївна, студентка гр. ПЗПІ-20-5, здобувачка вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Програмна система для організації практичних занять в автошколі. Back-end», що буде представлена до екзаменаційної комісії для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлена з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

Вступ 7
1 Аналіз предметної галузі 8
1.1 Аналіз предметної галузі 8
1.2 Аналіз ринку 9
1.3 Виявлення проблем та актуалізація рішень 17
1.4 Постановка задач 18
2 Перелік вимог до програмної системи 19
2.1 Постановка мети 19
2.2 Загальний опис системи 19
2.3 Основний функціонал системи 20
2.4 Загальні обмеження 22
2.5 Припущення та залежності 22
3 Архітектура програмного забезпечення 23
3.1 Проєктування архітектури програмної системи 23
3.2 Проєктування бази даних 24
3.3 Прецеденти використання системи 26
3.4 Архітектура серверної частини 29
4 Опис прийнятих програмних рішень 33
4.1 Вибір засобів програмної реалізації 33
4.2 Опис серверної частини системи 35
5 Тестування розробленого програмного забезпечення 44
5.1 Обґрунтування вибору виду тестування 44
5.2 Опис тестування 44
Висновки 48
Перелік джерел посилання 49
Додаток А 51

ВСТУП

Свобода пересування та незалежність від громадського транспорту є величезною перевагою та навіть необхідністю у сучасному світі. Для здобуття такої привілеї необхідно здати теоретичний та практичний екзамени з водіння, а навчитися цьому можливо в автошколі. За статистикою головного сервісного центру МВС України тільки в 2023 році посвідчення водія було видане вперше 337 278 особам [1]. Через війну в країні, наприклад, збільшилась кількість жінок, які здають на водійське посвідчення, тому що багато чоловіків захищають нашу країну, тому навичка керування автомобілем стає для них необхідністю.

Під час аналізу веб-сайтів популярних автошкіл було виявлено, що вони переважно пропонують лише інформаційні матеріали та майже не мають інтерактивних елементів. Зокрема, для управління розкладами занять часто використовуються звичайні паперові зошити, доступ до яких обмежений лише для інструкторів та персоналу автошколи. Цей підхід не лише не є надійним, але й не забезпечує зручності для учасників навчального процесу.

Відсутність інтерактивних інструментів та обмежений доступ до розкладів та інших важливих даних створює потребу у впровадженні нових технологій для оптимізації навчального процесу в автошколах. Враховуючи великий попит на отримання водійських посвідчень, постійний розвиток та зростання кількості автошкіл в Україні, а також застарілі методи ведення навчального процесу, стає очевидною необхідність у впровадженні програмного забезпечення для зручної організації занять у автошколах.

Метою роботи є створення серверної частини комплексної програмної системи для організації практичних занять в автошколі, яка надасть зручний інтерфейс для запису на заняття, відстеження прогресу та загалом підвищить якість навчального процесу під час проходження практичного курсу з водіння.

Під час виконання роботи було спроектовано та розроблено програмну систему, яка покращить процес організації занять в автошколі для інструкторів та для студентів. Для використання системи було розроблено веб-застосунок та мобільний додаток.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

Важливість мати водійське посвідчення і вміти добре керувати автомобілем складно переоцінити у сучасному світі. Вміння водити може значно спростити повсякденну діяльність людини, звільняючи від залежності від громадського транспорту та від погодних умов. Особливо у воєнний час, коли загроза небезпеки може з'явитися неочікувано, і вміння водити автомобіль може врятувати життя. У багатьох сім'ях традиційно чоловіки виконували роль водіїв, оскільки саме вони зазвичай були відповідальними за керування автомобілем. Проте в період війни багатьом з них довелося піти на фронт, що залишило сім'ї без водія. Відтак, зросла тенденція отримання водійських прав серед жінок може пояснитися не лише соціальними та економічними факторами, але й потребою в самозахисті та забезпеченні безпеки себе та своїх близьких у непередбачуваних умовах воєнного часу.

Окрім очевидного комфорту та зручності, вміння водити автомобіль є ключовим елементом особистої та професійної мобільності. Водійське посвідчення відкриває двері до широкого спектра можливостей, починаючи від забезпечення робочих можливостей та соціальної активності й закінчуючи підвищенням незалежності та особистої свободи. Будучи водієм, людина може легко переміщуватися за потребами роботи, навчання чи розваг, а також ефективно виконувати обов'язки у сфері побуту та сімейного життя.

Однак, з великою силою приходить і велика відповідальність, оскільки правильне володіння автомобілем вимагає уважності, дисципліни та відповідальності за свої дії на дорозі. Автомобільна промисловість постійно розвивається, вводячи нові технології та вдосконалюючи автомобільні системи для полегшення керування. Наприклад, компанія Tesla впроваджує концепцію повного самостійного керування (Full Self-Driving Capability), яка передбачає, що автомобіль зможе самостійно їздити практично будь-де з мінімальним втручанням водія [2]. Але попри такий технологічний прогрес, важливо пам’ятати, що відповідальним за дорожню ситуацію залишається водій. Вміння керувати

автомобілем, правильно реагувати на різні дорожні ситуації та дотримуватися правил дорожнього руху є необхідними для забезпечення безпеки на дорозі та запобігання дорожньо-транспортним пригодам.

Саме тому постає важливість автошкіл, які повинні надавати ефективну систему навчання, що дозволить максимально підготувати водіїв до реальних дорожніх умов. Розробка програмної системи, яка сприятиме організації практичних занять в автошколі, є важливим кроком у підвищенні якості навчання водінню та забезпеченні безпеки на дорогах.

1.2 Аналіз ринку

Аналіз конкурентів на ринку є важливою стратегією для розвитку програмного продукту. Він дозволяє збирати цінну інформацію про те, як конкуренти привертають та утримують своїх клієнтів. Це допомагає ідентифікувати успішні практики та уникнути помилок. Крім того, аналіз конкурентів дозволяє виявити ніші та можливості для інновацій, які можуть допомогти створити унікальну пропозицію для клієнтів та отримати перевагу над конкурентами. Саме тому під час аналізу ринку були вивчені декілька веб-сайтів популярних автошкіл України, з увагою на їх можливості та функціонал.

Першим предметом для аналізу є веб-сайт автошколи «Всеукраїнська автошкола онлайн» (див. рис. 1.1). Головна сторінка зустрічає користувачів з різноманітною інформацією про школу, її досягнення, тарифи та ціни, розкладом груп, описом кожного члена команди, відгуками, відповідями на поширені запитанні та, на останок, формою для залишення заявок. Окрім цього, на сайті представлено головне меню, яке включає такі пункти як: “Про нас”, “Послуги та ціни”, обране місто, “Відгуки” та “Контакти”.

Форма для залишення заявки з’являється не тільки у кінці головної сторінки а й при натисканні таких кнопок як “Записатися на навчання”, “Залишити заявку” або “Записатися”. Для того, щоб з користувачем зв’язалися працівники автошколи, потрібно залишити своє ім’я, номер телефону, не обов’язково нікнейм у Telegram та промокод, якщо такий є.

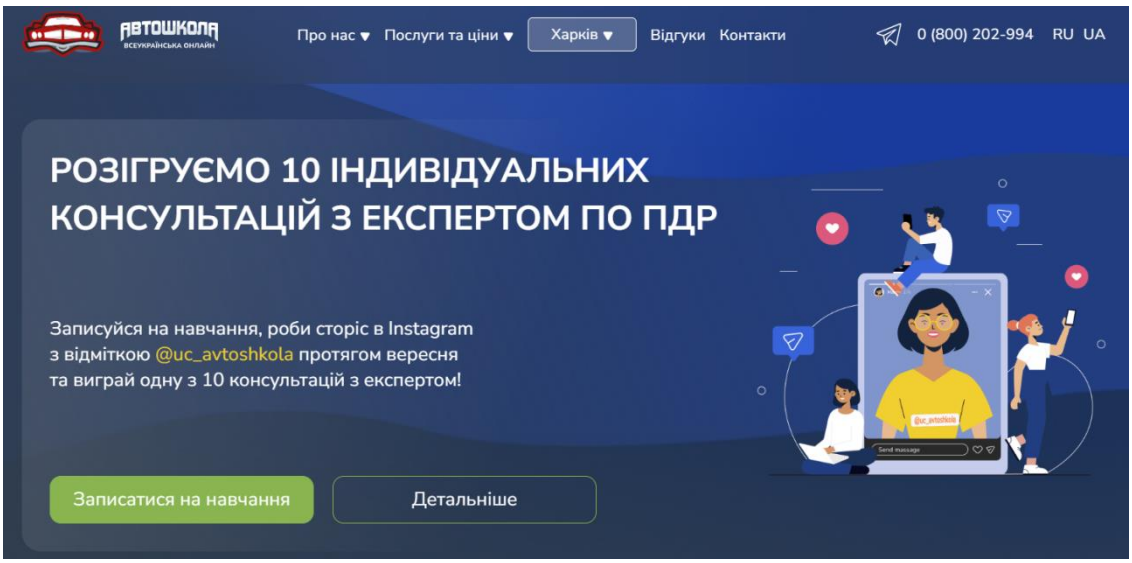


Рисунок 1.1 – Головна сторінка веб-сайту автошколи «Всеукраїнська автошкола онлайн»

На головній сторінці або в розділі “Послуги та ціни”>”Автошкола онлайн” є можливість передивитись та обрати бажаний тариф для вивчення теоретичної частини, а також оплатити частинами, через monobank або ПриватБанк, або одним платежем. Але при оплаті частинами, після відправлення форми з вашим ПІБ та номером телефону, замість сторінки з оплатою, користувача повідомляють про дзвінок працівника автошколи.

Зовсім інша історія трапляється, якщо обрати оплату одним платежем. В цьому разі користувача дійсно перекидає на сторінку з оплатою платіжного сервісу Fondy, де можна ввести дані банківською картки або оплатити через GooglePay.

Також на головній сторінці є опис поточних груп, в які проводиться набір учнів. Користувач може дізнатися розклад та старт занять, кількість залишених місць.

Наступний розділ “Послуги та ціни” також є випадаючим списком і пропонує користувачу всі послуги, які може надати ця автошкола. Він має такі пункти як: “Заняття з інструктором з водіння”, “Корпоративне навчання”, “Дитяча автошкола”, тощо. Більшість сторінок має інформаційно-описовий характер без наявності нового функціоналу.

У розділі “Дитяча школа” користувачам пропонується Telegram бот для батьків “Дитяча онлайн автошкола”, а також інтегрована гугл-форма, через яку представник школи може доєднати свою школу до занять автошколи.

Взагалі платформа автошколи надає 2 Telegram бота для спілкування користувачів з консультантами, другим з яких є бот однойменний назві автошколи. Використовуючи такий засіб спілкування, користувачі можуть задати будь-які питання та отримати відповіді від представників автошколи.

Наступний підрозділ “Заняття з інструктором з водіння” надає можливість переглянути інструкторів у обраному місту, їх машини, а також ціну заняття за 60 хв. Кнопка “Записатися на керування” перенаправляє до віджета програми для запису клієнтів EasyWeek. Тут можна обрати доступного інструктора, побачити його машину та ціну за заняття. Потім потрібно обрати дату і час заняття (див. рис. 1.2).

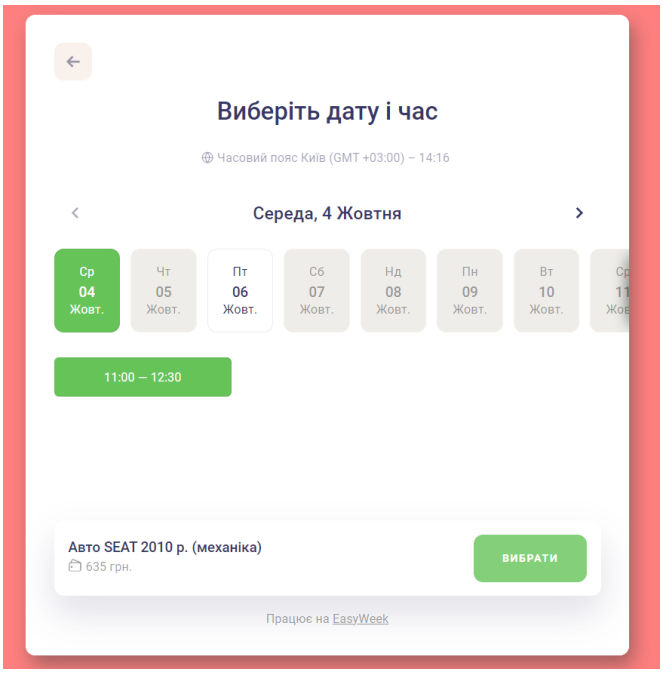


Рисунок 1.2 – Вибір часу та дати заняття через EasyWeek

Для того, щоб оформити запис потрібно ввести свої особисті дані, такі як: номер телефону, ім’я, прізвище, електронну пошту. Після відправки форми можна передивитись деталі бронювання, додати подію в календар або зробити ще один запис. Оплата заняття буде проходити безпосередньо після проведення заняття,

тож сайт не дає можливості оплатити його заздалегідь. Також через платформу EasyWeek можна скасувати урок.

Запис на заняття через сторонню платформу це непогано, якби не один серйозний недолік. На етапі вибору інструкторів завжди стоїть місто Дніпро, незважаючи на обране місто на сайті автошколи. Змінити місто на EasyWeek неможливо, як і записатися до інструктора не в Дніпрі.

Також хочеться зазначити, хоч це і проблема фронт-енду, якщо користувач знаходиться не на сторінці з інструкторами, то при виборі міста, взагалі не ясно що саме змінюється і може скласти враження відсутності функціоналу.

У розділі з відгуками сайт пропонує переглянути заготовлені відгуки певної кількості, в тому числі від зірок та блогерів. Залишити свій відгук не є можливості. Через такий підхід у користувача може скластися враження підроблених наявних відгуків.

Підсумувавши все вище сказане, можна зробити деякі висновки. До основних переваг сайту можна віднести:

- наявність наглядного опису тарифів та груп;
- можливість сплатити теоретичний курс одним платежем;
- наявність двох Telegram ботів для швидкого та зручного спілкування користувачів із представниками автошколи;
- можливість передивитись наявних інструкторів та їх машин відповідно до обраного міста;
- можливість записатися до обраного інструктора у певний день та час.

З недоліків можна відмітити:

- немає можливості оплатити теоретичний курс через сайт частинами, лише залишити заявку;
- немає можливості залишити власний відгук та передивитись не спеціально підготовлені відгуки звичайних людей, через що може скласти враження підробки;
- немає відгуків на кожного з інструкторів;
- немає можливості сплатити практичне заняття через сайт;

– немає можливості записатись до занять інших інструкторів, які знаходяться не в Дніпрі.

Наступним конкурентом для аналізу є веб-сайт автошколи «Персей» (див. рис. 1.3). На головній сторінці сайту автошколи «Персей» можна побачити меню, яке має наступні розділи: «ПДР», «Тести», «Відгуки», «Контакти», а також кнопки «Записатись», «Вхід», «Дивитись». Розглянемо функціонал кожного з розділів.

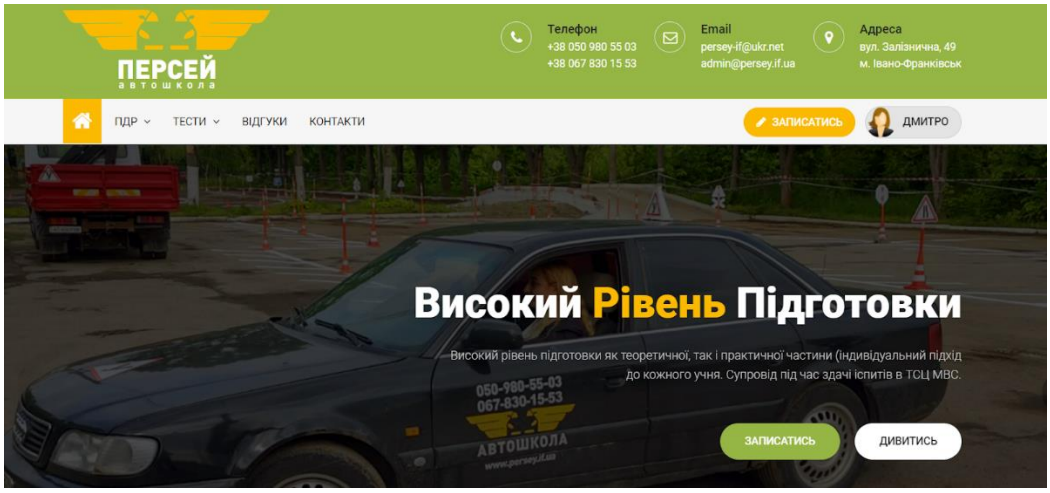


Рисунок 1.3 – Головна сторінка веб-сайту автошколи «Персей»

Розділ «ПДР» являє собою випадаючий список, що має такі підрозділи як: «Правила дорожнього руху», «Дорожні знаки та розмітка», «Перша медична допомога». Кожен підрозділ включає в собі список відповідних категорій, які надають детальну інформацію щодо теми.

Розділ «Тести» ділиться на 2 підрозділи: «Навчання» та «Екзамен». Навчальні тести можна обирати споміж 24 тем, без обмежень у часі, але без підписки доступ відкрито тільки до 15 питань на тему за 1 спробу. Також є можливість використати підказку, але кількість використаних підказок впливає на загальний рейтинг результатів. Після надання відповіді на запитання одразу можна подивитись чи була вона правильною, пояснення щодо відповіді, прогрес та поточний результат тесту у відсотках.

При завершенні тесту, навіть не відповівши на всі запитання, результат тесту зберігається. Всі результати тестів можна подивитись у профілі користувача

в розділі «Результати тестів» з можливістю продовжити тест, почати новий, або переглянути відповіді (див. рис. 1.4).

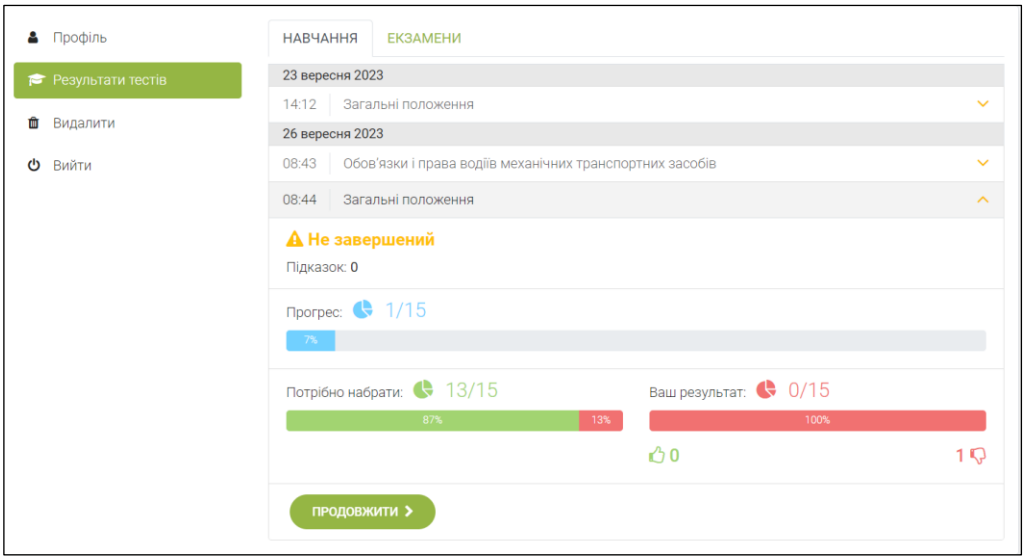


Рисунок 1.4 – Перегляд результатів тесту

Важливо зазначити, що під підпискою мається на увазі не реєстрація на сайті або оплата кожного місяця, а саме офіційне зарахування учня в цій школі. Причому, за відгуком одного з учнів автошколи, робиться це не через панель адміністратора на сайті, а напряму, змінюючи відповідне поле у СУБД, що є значним недоліком системи через підвищений ризик помилки та пошкодження даних.

В наступному розділі «Відгуки» є можливість додати відгук, відповісти або цитувати інший відгук. При додаванні відгуку потрібно також заповнити поля для імені та теми, тобто будь-який гість сайту може залишити коментар без реєстрації та підписки. Мінусом такого підходу є можливість напливу негативних відгуків від конкурентів, тому що цей процес ніяк не контролюється на стороні серверу. Окрім того, користувач може оцінити відгук поставивши лайк або дизлайк, але цей функціонал працює не досконало. По-перше, зробивши вибір, змінити його неможливо. По-друге, інколи збільшення кількості лайків або дизлайків була некоректною, оскільки замість 1 кількість збільшувалась на 2.

Кнопка «Записатись» відповідає за запис на курси водіїв. Перейшовши на сторінку, потрібно заповнити форму з особистими даними. Якщо зареєстрований

користувач заповнює форму, то такі дані як номер телефону, ім’я та електронна пошта одразу підтягуються.

Для реєстрації, потрібно натиснути на кнопку “Вхід” і потім обрати вкладку «Реєстрація», де потрібно заповнити форму, а саме: ім’я, номер телефону, електронну пошту, пароль та його підтвердження. Після успішної реєстрації, з’являється профіль користувача, де можна подивитись або змінити свої дані, додати фотографію після підтвердження пароля. Але змінити пароль, на жаль, неможливо.

З головних плюсів даного веб-сайту можна відзначити:

– збереження результатів проходження тестів;
– наявність великої кількості тестів та теоретичного матеріалу.

З основних мінусів можна виділити:

– відсутність панелі адміністратора;
– неконтрольований процес залишання відгуків, а саме те, що будь-який гість сайту може залишити відгук з вигаданим ім’ям;
– відсутність будь-якого функціоналу пов’язаного з практичними заняттями та інформацією щодо інструкторів, тобто на чому базується розроблювана програмна система.

Останнім конкурентом є веб-сайт автошколи «Навігатор» (див. рис. 1.5). На головній сторінці автошколи «Навігатор» користувачу надається загальна інформація про школу, її акції та новини, філіали, інструкторів, тощо.

В першу чергу в очі кидається кнопка «Записатись», яка перенаправляє до форми запису. Там користувачу потрібно записати своє ім’я, номер телефону, обрати автошколу, час занять, програму навчання та додаткові послуги. За бажанням можна залишити коментар.

Кожному з розділів головного меню в основному притаманний інформаційно-описовий характер, в яких розповідається про особливості автошколи по даній темі, розклади, контакти, і таке інше.

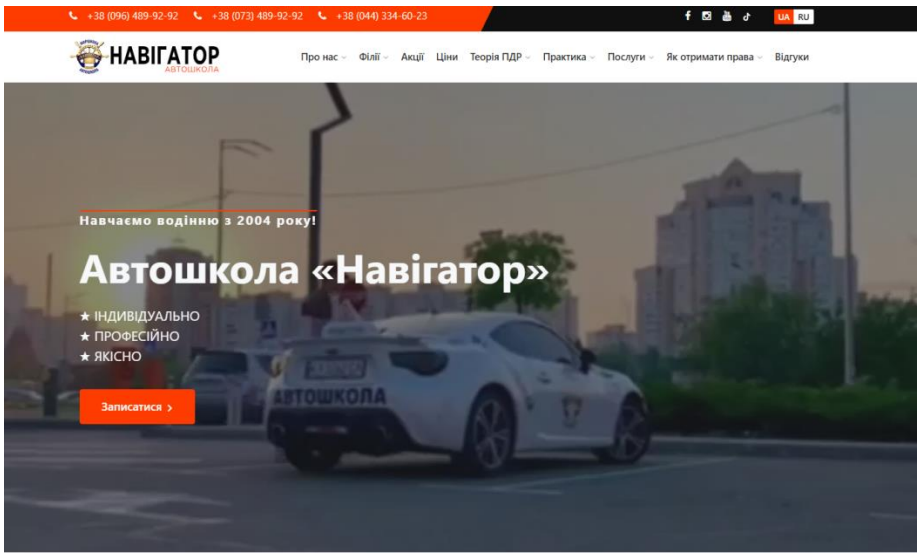


Рисунок 1.5 – Головна сторінка веб-сайту автошколи «Навігатор»

В розділі “Про нас”>”Наші інструктори” можна переглянути список наявних інструкторів в автошколі. Також можна побачити їх рейтинг і кількість людей, що залишили їм відгук.

Фотографія інструктора є одночасно і посиланням, тож при натисканні користувача перенаправляє на окрему сторінку з детальною інформацією про інструктора: в якій філії він викладає, на якому авто будуть проводитися заняття, тощо (див. рис. 1.6). Окрім того, на кожного інструктора є відгуки. Для того, щоб залишити відгук сперш потрібно авторизуватися, що можна зробити за допомогою Facebook або Google акаунту.

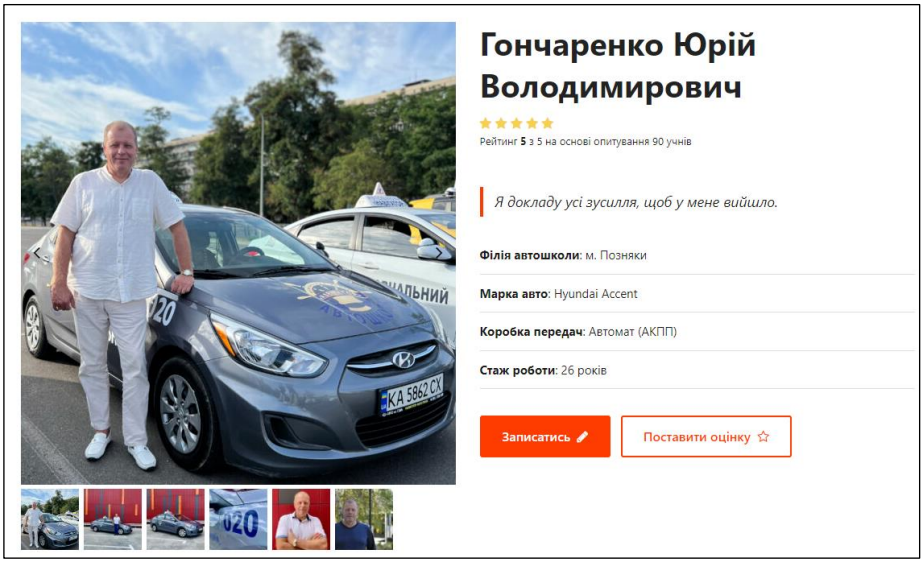


Рисунок 1.6 – Детальна інформація про інструктора

Для швидкого і зручного спілкування з представниками автошколи, веб-сайт втілив онлайн-чат, де користувач може задати питання консультантам.

Детально розглянувши даний веб-сайт, перейдемо до висновків. З плюсів можна відзначити такі риси:

- є можливість передивитись інструкторів та інформацію про них;
- є можливість залишити та продивитись відгуки на конкретного інструктора;
- для того, щоб залишити відгук потрібно авторизуватися за допомогою акаунту Facebook або Google, що мінімізує ризик напливу поганих відгуків від конкурентів;
- присутній онлайн-чат для швидкого спілкування з консультантами автошколи.

До мінусів веб-сайту належать:

- відсутність можливості записатися на практичне заняття до обраного інструктора;
- відсутність можливості сплатити заняття;
- при першому відкритті веб-сайту, сторінки та фотографії завантажуються достатньо довго.

1.3 Виявлення проблем та актуалізація рішень

Проаналізувавши декількох конкурентів на ринку, можна зробити висновок, що більшість з них надають лише інформаційні послуги, а з функціоналу присутня лише форма для залишення заявки. Деякі з них фокусуються більше на теоретичній частині, пропонуючи користувачам свій веб-сайт як платформу для вивчення матеріалу і проходження тестів. Лише одиниці мають дійсно корисний функціонал, спрямований на практичні заняття, оплату, тощо. Навіть у таких випадках, як правило, використовуються сторонні платформи, що може призвести до зниження контролю над процесом і збільшення ймовірності помилок. Тому можна зробити висновок, що повноцінних конкурентів нашій програмній системі наразі просто немає.

1.4 Постановка задач

Головною задачею є розробка комплексної програмної системи, а саме серверної частини, веб-сайту, та мобільного додатку, для організації практичних занять в автошколі, яка буде надавати функціонал як для студентів, інструкторів, так і адміністраторів автошколи.

Користувачі, які захочуть стати студентами автошколи, зможуть переглядати інструкторів та їх машини, а також подати заявку для того, щоб закріпитись за обраним інструктором. Адміністратор автошколи потім зможе переглядати заявки та приймати або відхиляти їх. Також користувачі зможуть переглядати відгуки та рейтинг інструктора та машини. Залишати відгуки інструктору зможуть лише студенти, які навчались в цього інструктора, таким чином процес залишання відгуків буде контрольованим. Окрім цього, студенти зможуть підтримувати зв’язок з їх інструктором за допомогою чату, якщо буде така потреба. Але чи не найголовнішим функціоналом платформи буде можливість переглядати розклад інструктора, записуватись на заняття в обрані день та час та оплачувати заняття онлайн. У своєму профілі студент зможе знайти чек-лист практичних вмінь, які студент має опанувати до іспиту, таким чином відслідковуючи свій прогрес.

З іншого боку, інструктор може зручно переглядати власний графік занять та відстежувати, на яку годину студент записаний, а також використовувати чат для комунікації з ним.

Як вже зазначалось раніше, система буде підтримувати механізми для адміністрування, як створення профілів інструкторів та їх машин, приймання або відхилення заявок студентів, позначення студентів, які закінчили навчання в автошколі, неактивними тощо.

2 ПЕРЕЛІК ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Постановка мети

Метою виконання роботи є сервера частина комплексної програмної системи для організації практичних занять в автошколі, яка дозволить студентам та інструкторам більш зручно взаємодіяти між собою, записуватися та сплачувати заняття та загалом покращить враження від навчального процесу. Система повинна бути надійною, забезпечуючи стабільну та безперебійну роботу, щоб користувачі могли надійно скористатися всіма її функціями.

Для реалізації серверної частини використовувалась мова програмування JavaScript, платформа Node.js з використанням фреймворку Express. Для зберігання фотографій було обрано сервіс зі зберігання об’єктів Amazon S3. У якості постачальника сервера електронної пошти для надсилання електронних листів користувачам був обраний сервіс SendGrid. Окрім цього, для зберігання даних було використано хмарну базу даних MongoDB Atlas, що забезпечить високий рівень доступності даних.

Оскільки програмна система складається з веб-застосунку, мобільного додатка, сервера та бази даних, вона використовує трирівневу архітектуру з клієнт-серверною моделлю. Веб-застосунок і мобільний додаток виступають у ролі клієнтів, які звертаються до сервера для отримання та обробки даних. Сервер є відповідальним за обробку запитів від клієнтів, взаємодію з базою даних та відправку відповідей назад клієнтам. Така архітектура забезпечує чітке розділення функціональності та полегшує розробку, масштабування та підтримку системи, оскільки окремі компоненти можуть бути незалежно розвивати та модифікувати.

2.2 Загальний опис системи

Програмна система являє собою клієнт-серверний додаток, що має у складі веб-застосунок та мобільний додаток.

Система дозволить користувачам стати студентами автошколи. Студенти зможуть зручно переглядати всіх інструкторів автошколи, їх машину та детальну інформацію про них. Після вибору інструктора, студент може подати заявку на

закріплення за уподобаним інструктором. З іншого боку, адміністратор автошколи зможе переглядати усі заявки і профілі студентів та вирішувати прийняти чи відхилити заявку. Важливим аспектом програми є можливість переглянути розклад інструктора, обрати вільний день та час та записатися на заняття. Також користувачі зможуть оплатити заняття декількома способами: за кожен урок окремо або використати баланс занять акаунту. Оплата буде проходити за допомогою платіжного сервісу LiqPay. За бажанням студент зможе додати урок до свого Google календарю, або відмінити за 24 години. Не менш важливим є можливість спілкування між студентом і його інструктором у вбудованому чаті, якщо буде така потреба. Окрім цього, у власному профілі студент може знайти чек-лист вмінь, якими необхідно йому оволодіти до проведення іспиту, що допоможе відслідковувати прогрес та розумітися на своїх слабких і сильних сторонах.

Звісно, що система приносить користь й інструкторам також. За допомогою неї інструктор може зручно переглядати свій розклад у будь-який момент, знаючи точно хто зі студентів на яку годину в нього записаний. Також додаток пропонує контрольовану систему відгуків та рейтингу, де залишати відгук про інструктора може лише студент, який в нього навчався, забезпечуючи таким чином достовірність і показовість коментарів.

Також система доступна для використання адміністраторами автошколи. Вони будуть відповідальні за маніпулювання даними, наприклад як створення профілів інструкторів та їх машин, приймання або відхиляння заявок студентів.

2.3 Основний функціонал системи

Програмна система розрізняє 3 типи користувачів – Student, Instructor, Administrator. Кожна роль має доступ до певного набору функціоналу.

Функціонал, який доступний всім користувачам незалежно від ролі:

- можливість авторизуватись в системі;
- можливість змінити пароль;
- можливість відновити пароль, якщо забув його;

– перегляд всіх інструкторів з можливістю їх фільтрації та сортування;
– окремий перегляд інструктора з більш детальною інформацією;
– перегляд відгуків та рейтингів інструктора та автомобіля;
– перегляд розкладу інструктора.

Програмна система надає наступний додатковий функціонал користувачу з роллю Student:

– можливість зареєструватися в системі;
– можливість переглядати та редагувати свій профіль;
– подання заявки на закріплення за інструктором;
– запис на заняття до інструктора в обрані день та час;
– перегляд інформації про свої заняття;
– можливість додати заняття у свій Google календар;
– можливість відмінити заняття;
– можливість оплатити заняття декількома способами;
– можливість залишити відгук про інструктора;
– можливість поставити оцінку інструктору та його автомобілю;
– можливість спілкування з інструктором у чаті;
– ведення чек-листу вмінь.

Додатковий функціонал для користувача з роллю Instructor:

– можливість переглядати свій профіль;
– можливість спілкування зі студентами у чаті;
– перегляд власного розкладу з інформацією про студентів;
– відміна заняття з розкладу.

Наступний додатковий функціонал система пропонує користувачам із роллю Administrator:

– управління профілями інструкторів та їх машинами;
– перегляд даних про всіх студентів;
– перегляд та приймання або відхиляння заявок від студентів;
– відміна заняття з розкладу.

2.4 Загальні обмеження

Обмеження програмної системи включають:

- для функціонування системи необхідне підключення до мережі Інтернет;
- адміністратор не може змінювати персональні дані студентів;
- кожен користувач системи повинен мати унікальну поштову адресу;
- студент не може записатись на заняття, якщо він не закріплен за інструктором;
- студент може залишити відгук та поставити оцінку тільки своєму інструктору;
- студент може спілкуватися у чаті тільки зі своїм інструктором, а інструктор тільки зі своїми студентами.

2.5 Припущення та залежності

Система має наступні припущення та залежності:

- для належної роботи веб-застосунку необхідна підтримка браузером HTML5 та CSS3;
- Мобільний додаток можна використовувати лише на пристроях з операційною системою Android.
- сервіси MongoDB Atlas та Amazon S3 працюють цілодобово і безперервно.

3 АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Проєктування архітектури програмної системи

Архітектура розроблюваної програмної системи є комплексною і багаторівневою, а саме вона базується на розподіленій клієнт-серверній архітектурі. Такий підхід дозволяє розділити функціонал клієнта, з яким безпосередньо взаємодіє кінцевий користувач, та сервера, який відповідає за всю бізнес-логіку додатку. У даному випадку, маємо 2 клієнти: є веб-застосунок та мобільний додаток, які виконуються на пристроях користувачів. На рисунку 3.1 зображено діаграму розгортання, що демонструє загальну схему спроектованої архітектури системи.

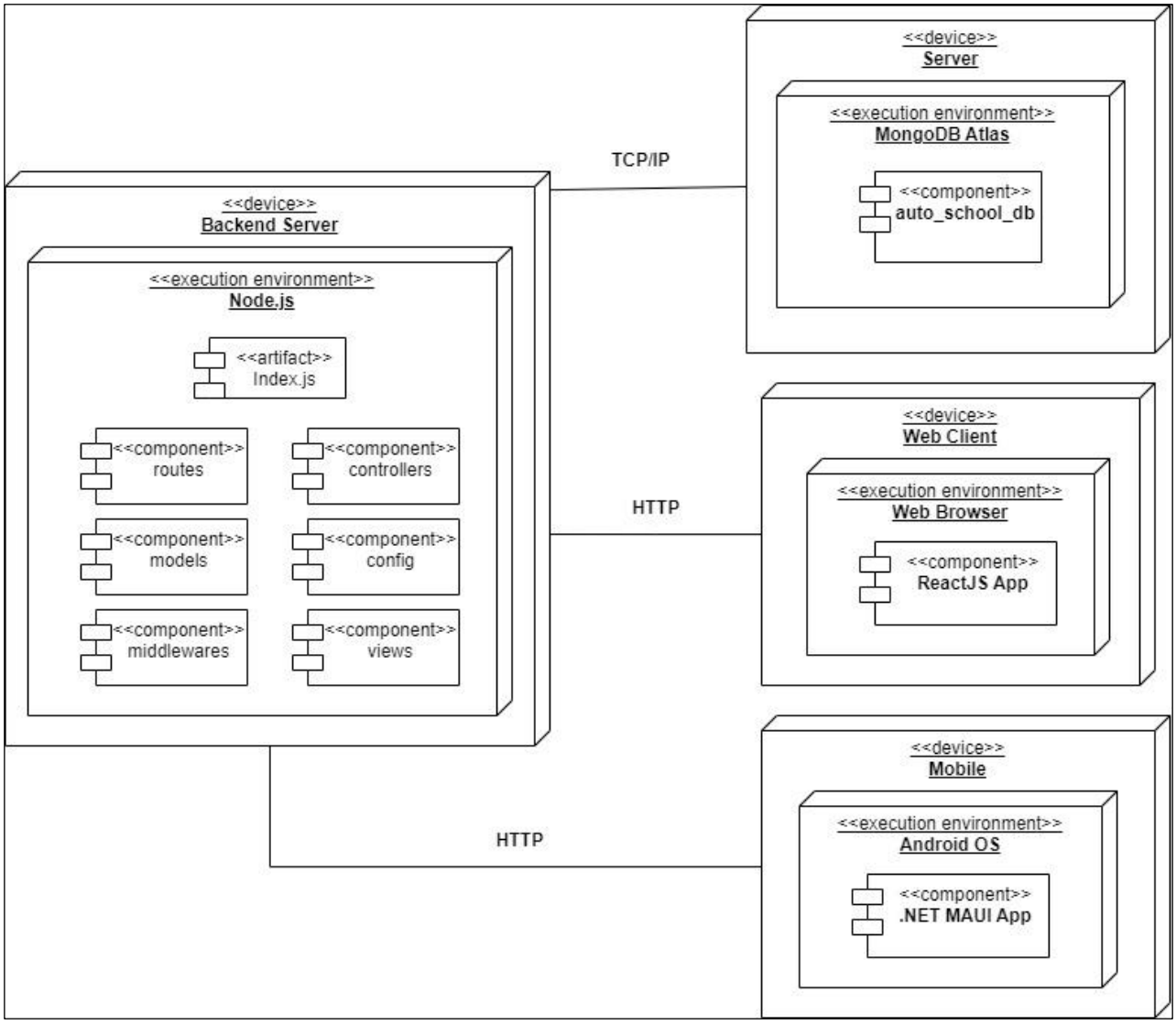


Рисунок 3.1 – Діаграма розгортання програмної системи (рисунок виконаний самостійно)

Для реалізації серверу була обрана мова програмування JavaScript та платформа Node.js з фреймворком Express. Серверна частина, яка реалізує принципи REST API, відповідає за обробку запитів, отриманих від клієнтів, виконання бізнес-логіки додатку, взаємодію з базою даних та іншими сервісами, а також процес автентифікації та авторизації користувачів. Клієнти, у свою чергу, відповідають за відображення інформації користувачу та обробку його взаємодії з інтерфейсом.

Архітектура системи є розподіленою, тому що зв'язок між клієнтом і сервером здійснюється через мережу Інтернет, де клієнти взаємодіють із сервером через HTTP запити.

Загалом така архітектура є стандартом для багатьох сучасних веб-та мобільних додатків, оскільки дозволяє ефективно вирішувати багато завдань розробки та взаємодії з користувачами.

3.2 Проєктування бази даних

MongoDB було обрано як базу даних з-за її численних переваг, які відповідають вимогам програмної системи. MongoDB, як документ-орієнтована база даних, дозволяє легко змінювати схему даних, що особливо корисно для швидкого розвитку та зміни вимог. Окрім того, однією з найважливіших переваг використання Node.js і MongoDB разом є їх нативна підтримка JSON (JavaScript Object Notation). Драйвер MongoDB для Node.js робить їх використання бездоганним. Драйвер автоматично зіставляє об’єкти JavaScript з документами BSON, що означає, що розробники можуть легко працювати з даними [3]. А також асинхронна архітектура Node.js ідеально поєднується з асинхронними драйверами MongoDB, забезпечуючи ефективну взаємодію з базою даних. Це сприяє покращенню масштабованості та пропускної здатності додатків, забезпечуючи відмінний користувацький досвід навіть при великому трафіку.

MongoDB Atlas, як керована хмарна послуга MongoDB, надає додаткові переваги, такі як автоматичне масштабування, резервне копіювання даних та

високий рівень доступності. Це забезпечує надійність та стабільність роботи системи, а також полегшує адміністрування та моніторинг бази даних.

На рисунку 3.2 зображено схему бази даних програмної системи.

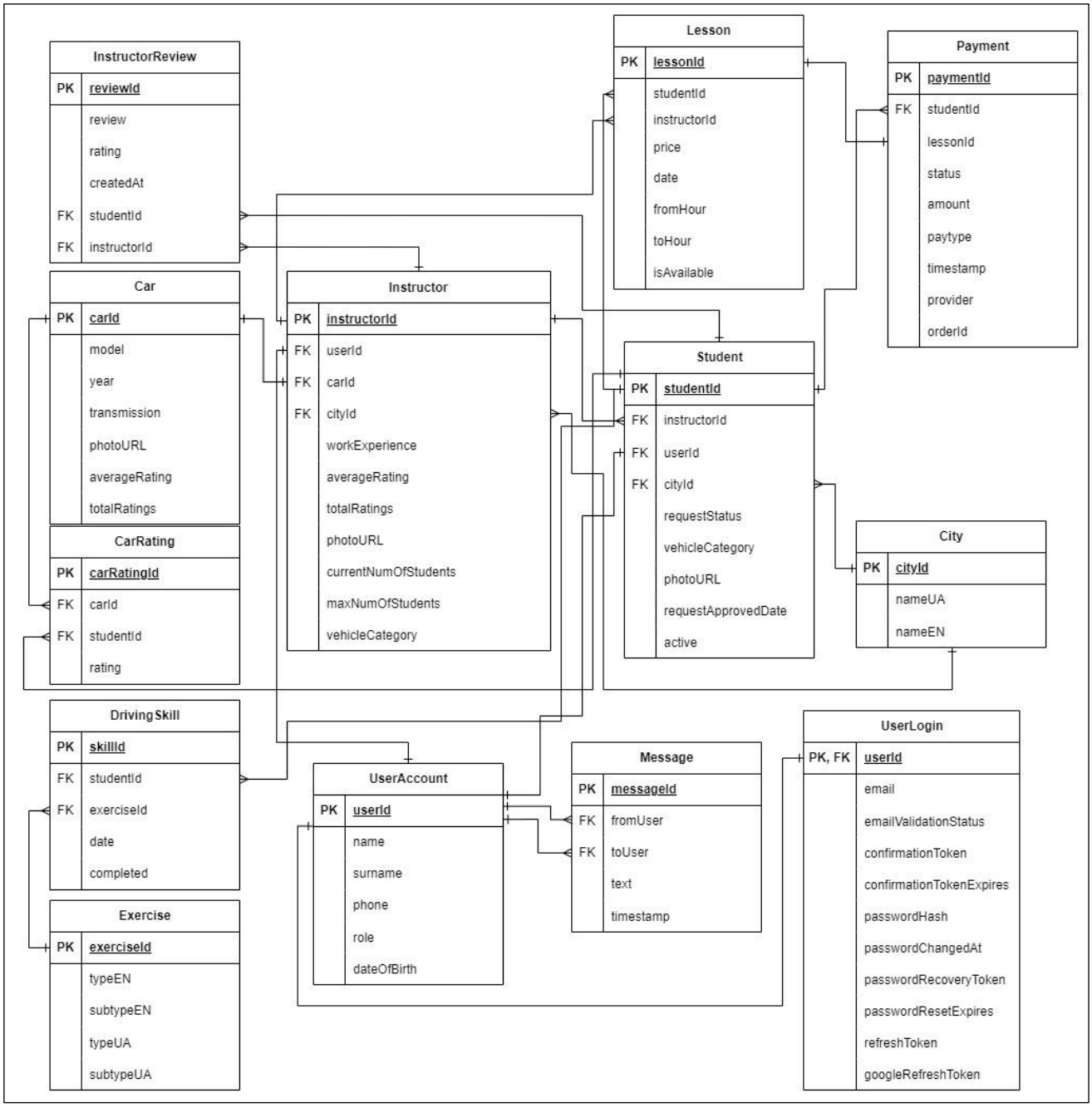


Рисунок 3.2 – ER-Діаграма бази даних (рисунок виконаний самостійно)

Зі схеми можна побачити, що вона є доволі комплексною, включає в собі багато сутностей та різноманіття зв’язків. Зокрема в базі даних будуть зберігатися дані про користувачів, профілі інструкторів та студентів, автомобілі, відгуки, заняття та оплату, дані необхідні для авторизації та автентифікації тощо.

3.3 Прецеденти використання системи

На етапі планування розробки програмної системи для організації практичних занять в автошколі були визначені основний функціонал системи та створені діаграми прецедентів, які дозволяють покати типи ролей та їх взаємодію з системою. Такі діаграми створюються для простого пояснення роботи системи, створення функціональних вимог, основ для тестування тощо [4].

Користувачі мають різний доступ до функціоналу, тому у системі передбачено розділення користувачів за ролями: студент, інструктор та адміністратор. Система також надає певний функціонал і для незареєстрованого користувача. Ролі надаються користувачам різними способами, наприклад, при реєстрації користувачу надається роль студента, а ролі інструктора та адміністратора надаються при створенні адміністратором відповідного користувача системи. На рисунку 3.3 зображено діаграму прецедентів для користувача з роллю студента.

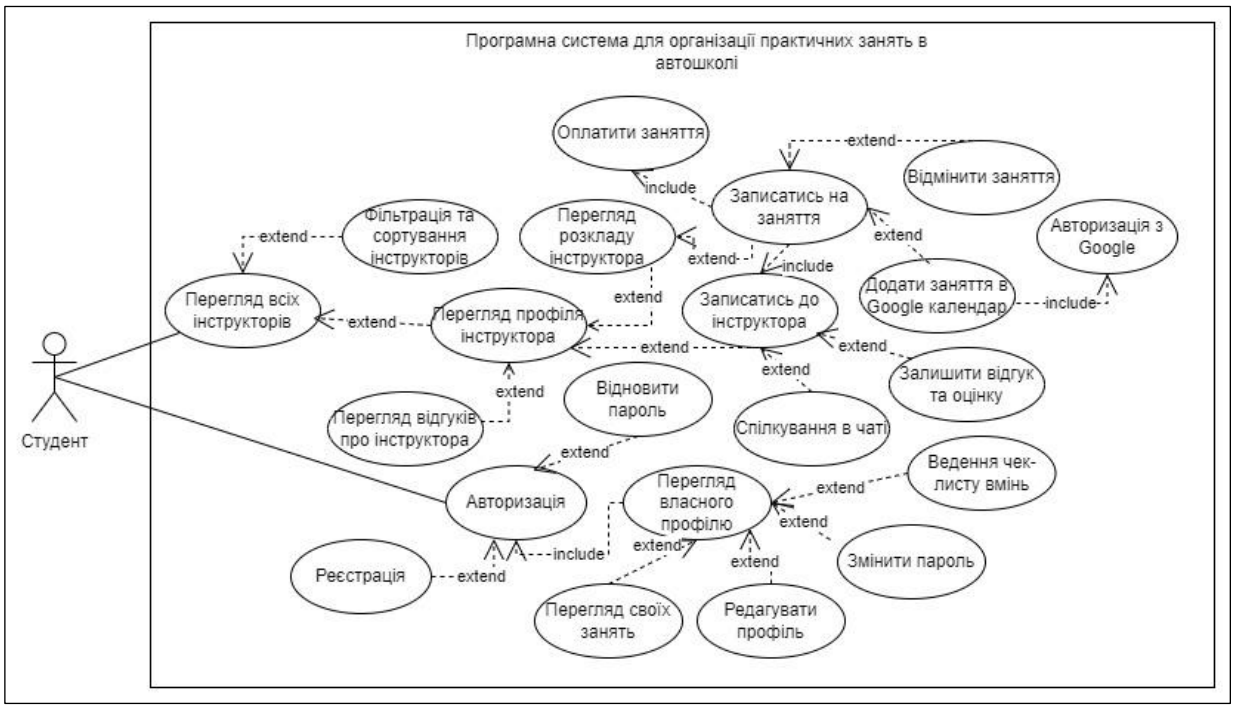


Рисунок 3.3 – Діаграма прецедентів для студента (рисунок виконаний самостійно)

Студент може авторизуватися за допомогою комбінації електронної пошти та паролю. Також є можливість відновити пароль, якщо користувач забув його.

Якщо студент ще немає власного акаунту в системі, для доступу до повного функціоналу, йому потрібно зареєструватися за допомогою електронної пошти, яка повинна бути унікальної в системі. Студент має змогу передивитися всіх інструкторів, фільтрувати та сортувати їх, а також побачити більш детальний опис кожного інструктора з його автомобілем та відгуками та розклад інструктора. Він може подати заявку на закріплення за інструктором, якщо він підтвердив свою електронну пошту та в інструктора ще є вільні місця. Закріплений за інструктором студент може записатися на практичне заняття до цього інструктора, а також оплатити його. Заняття можна додати до Google календарю, або відмінити його. Студент може подивитись всі минулі та поточні заняття. Окрім цього, він може залишити відгук та поставити оцінку своєму інструктору. Студент має свій профіль, в якому він зможе редагувати персональні дані, змінити пароль та вести чек-лист вмінь для відстеження прогресу. На останок, система дає можливість студенту спілкуватись зі своїм інструктором у чаті.

На рисунку 3.4 зображено діаграму прецедентів для інструктора.

Рисунок 3.4 – Діаграма прецедентів для інструктора (рисунок виконаний самостійно)

Інструктор, так само як і студент, може авторизуватися та відновити пароль, переглянути список всіх інструкторів, їх детальний опис та відгуки, фільтрувати та сортувати. Інструктор також має свій профіль, де він може змінити пароль, подивитись свій розклад з інформацією про студентів, відмінити заняття з розкладу за потреби, а також спілкуватися зі своїми студентами у чаті.

На рисунку 3.5 зображено діаграму прецедентів для адміністратора.

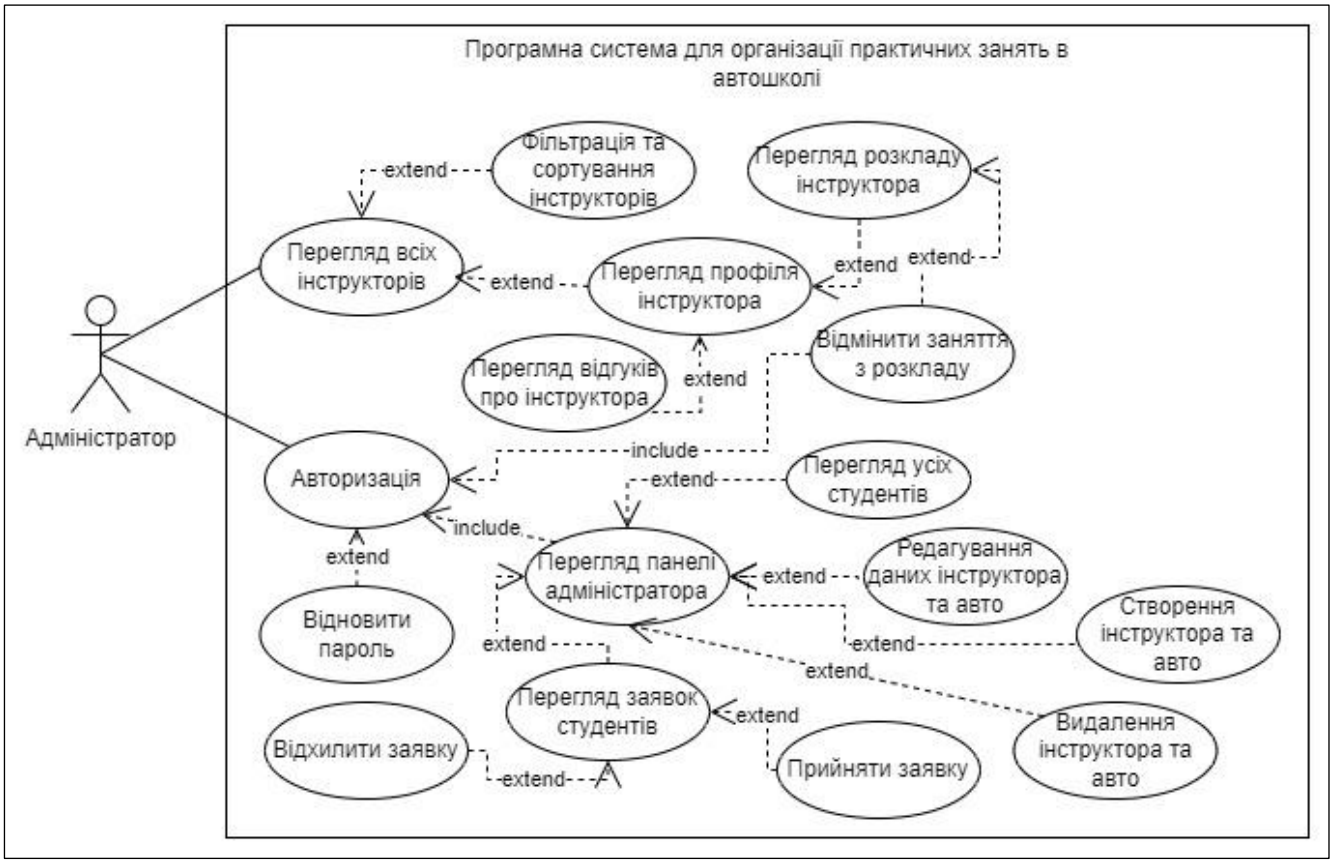


Рисунок 3.5 – Діаграма прецедентів для адміністратора (рисунок виконаний самостійно)

Адміністратор системи також може авторизуватися, відновити пароль, переглянути список всіх інструкторів, їх детальний опис та відгуки, фільтрувати та сортувати. Він може переглянути розклад інструкторів та відмінити заняття за потреби. Використовуючи панель адміністратора, він може переглянути всіх студентів. Також адміністратор може переглянути всі заявки студентів на закріплення за інструктором, прийняти або відхили заявку. Адміністратор також

створює інструкторів в системі разом з їх автомобілем, може редагувати та видаляти ці дані.

На рисунку 3.6 зображено діаграму прецедентів для незареєстрованих користувачів – гостей.

Рисунок 3.6 – Діаграма прецедентів для гостя системи (рисунок виконаний самостійно)

Гість системи може переглядати всіх інструкторів, фільтрувати та сортувати їх, переглядати їх детальний опис та відгуки, а також їх розклад. Звісно що гість може зареєструватися і отримати повний доступ до функціоналу системи як студент.

3.4 Архітектура серверної частини

При розробці серверної частини програмної системи була використана мова програмування JavaScript та фреймворк Express.js платформи Node.js. Розробка додатків на Node.js пропонує керовану подіями (event-driven) архітектуру, неблокуючий ввід-вивід і здатність ефективно обробляти кілька користувачів, що робить його висококонкурентним, особливо при інтеграції з MongoDB для створення високошвидкісних і масштабованих серверів баз даних [5]. Окрім цього, Node.js має екосистему з багатьма готовими модулями та можливість використання JavaScript як на клієнтській, так і на серверній стороні.

Для взаємодії з базою даної MongoDB була використана бібліотека Mongoose, яка спрощує роботу з базою даних, надаючи можливість визначення структури даних за допомогою схем, валідацію введених даних, побудову складних запитів за допомогою зручного API, підтримку різних типів даних та автоматичне створення індексів для оптимізації швидкості роботи з базою даних [6].

Архітектура серверної частини є багаторівневою, включаючи в себе маршрути (routes), контролери та моделі. На початку, маршрути відповідають за обробку вхідних запитів, визначаючи, які контролери повинні бути викликані. Контролери, у свою чергу, виконують логіку додатку та взаємодіють з базою даних через моделі, використовуючи дані із запиту. Моделі використовуються для представлення даних та взаємодії з базою даних, виконуючи операції читання, запису, оновлення та видалення даних [7].

Авторизація користувачів здійснюється за допомогою JWT-токенів, що гарантує безпеку та автентифікацію. Зокрема, використовуються два типи токенів: короткочасний access токен та довготривалий refresh токен. Зберігаючи довготривалий refresh токен у базі даних, система дозволяє користувачам оновлювати свій access токен без необхідності повторної авторизації кожного разу, коли він закінчується. Це забезпечує довготривалу сесію користувача та зручність використання системи [8]. Короткотривалість access токену необхідна з міркувань безпеки. З тієї ж причини обидва токени зберігаються на стороні клієнта у http-only cookie. Це означає, що вони доступні лише для серверних запитів і недоступні для JavaScript, що знижує ризик зловживання атаками міжсайтового скриптування (XSS). Такий підхід дозволяє забезпечити безпеку та конфіденційність даних користувачів, утримуючи токени в надійному середовищі, що не доступне для потенційних атак. На рисунку 3.7 зображено процес успішного входу користувача до системи.

Після отримання токенів, для кожного наступного запиту, який вимагає авторизованого користувача, клієнт прикріпляє до запиту http-only cookie. Сервер

проводить валідацію токену, перевіряючи підпис і декодуючи дані, щоб переконатися в автентичності та авторизації користувача [9].

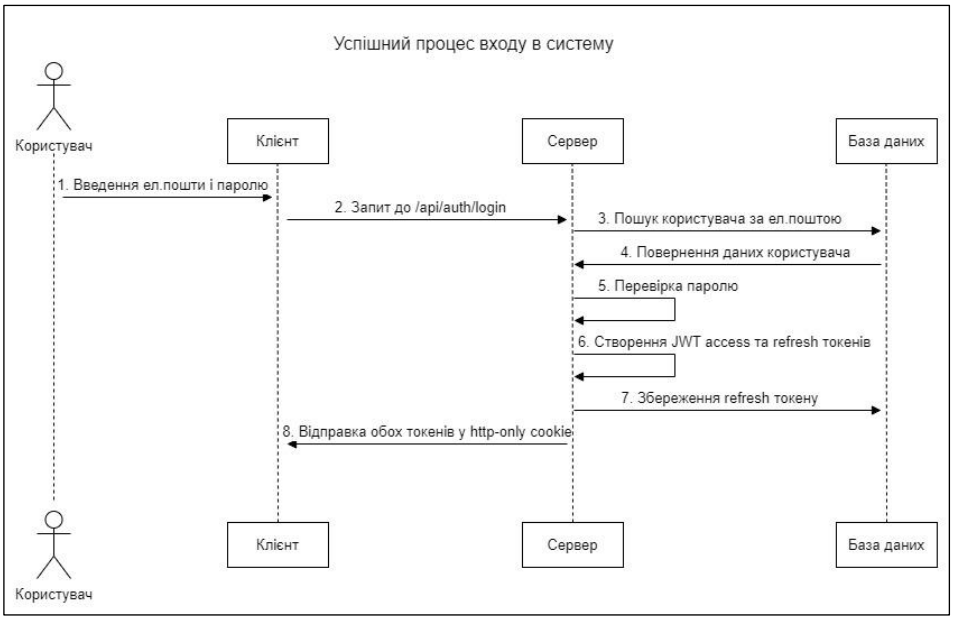


Рисунок 3.7 – Процес успішного входу в систему (рисунок виконаний самостійно)

Також сервер перевіряє роль користувача і чи має він право доступу до ресурсу. У разі успіху сервер відправляє клієнту запитуваний ресурс (див. рис. 3.8).

Рисунок 3.8 – Процес успішної авторизації користувача (рисунок виконаний самостійно)

Процес запису студентом на заняття продемонстровано на діаграмі діяльності, яка показує необхідні кроки та перевірки, які потрібно зробити, для того, щоб студент записався на заняття (див. рис. 3.9). Якщо студент хоче записатися на урок з водіння, перш за все він повинен бути закріплений за інструктором. У випадку коли студент ще не має свого інструктора, йому потрібно його обрати, таким чином надіславши запит на закріплення до адміністратора. Умовами закріплення до інструктора є підтверджена електронна пошта студента, а також наявних вільних місць у інструктора. Якщо всі умови були задовільні, студент може обрати день та час заняття та сплатити його, або використати баланс, якщо такий існує.

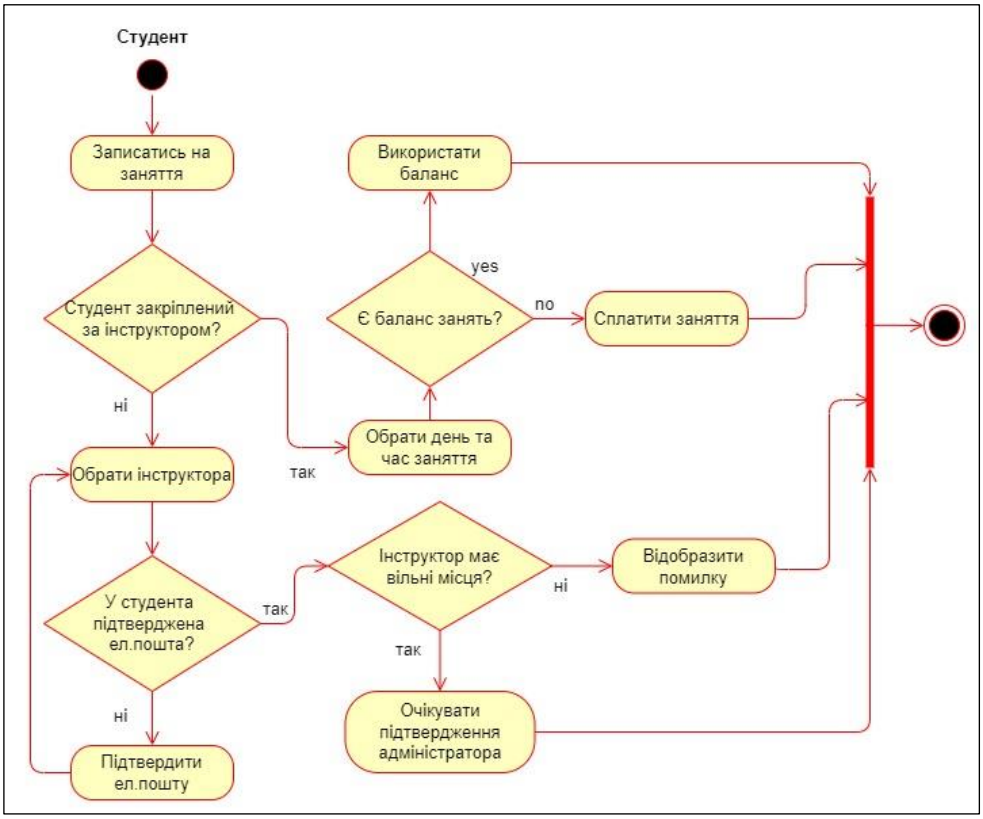


Рисунок 3.9 – Діаграма діяльності запису на заняття (рисунок виконаний самостійно)

Ця архітектура надійно захищає дані користувачів, забезпечуючи швидку та ефективну роботу системи. Кожен компонент відповідає за свою частину функціональності, що робить систему легко розширюваною та підтримуваною.

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Вибір засобів програмної реалізації

Програмна система для організації практичних занять в автошколі є комплексною системою, яка складається з серверної, мобільної та веб частин. Тому необхідно обрати такі технології та програмні засоби, які задовільнять потреби при розробці продукту, забезпечуючи швидку розробку функціоналу, а також надійний та легко масштабований програмний продукт. Тому для створення серверної частини системи було обрано мову програмування JavaScript у поєднанні з платформою Node.js та базою даних MongoDB Atlas.

JavaScript є однією з найбільш гнучких та потужних мов програмування для розробки серверних додатків. Завдяки платформі Node.js, JavaScript може використовуватися для побудови високопродуктивних, масштабованих серверів. Вона має асинхронну природу, що дозволяє ефективно управляти великими об'ємами одночасних запитів, що робить її ідеальним вибором для розробки веб-серверів з високим рівнем навантаження. Перевагами використання JavaScript для серверного програмування є швидкість розробки, оскільки вона є однією з найпопулярніших мов для роботи з клієнтською стороною веб-додатків, а також єдність мови між клієнтською і серверною стороною, що спрощує розуміння та підтримку коду. Додатково, JavaScript має велику кількість зовнішніх бібліотек та фреймворків, які значно спрощують розробку. Наприклад, Express.js - один з найпопулярніших фреймворків для побудови веб-додатків, який надає простий та елегантний спосіб створення серверної логіки.

Node.js – це вільна, відкрита платформа для виконання JavaScript на стороні сервера, заснована на двигуні V8 від Google. Вона відкриває двері для розробки серверних додатків за допомогою JavaScript, яка традиційно використовувалася лише для клієнтської частини веб-додатків. Однією з ключових характеристик Node.js є асинхронна модель введення/виведення, що дозволяє ефективно обробляти багатопотокові операції без блокування і використання великої кількості одночасних підключень. Крім того, вбудований двигун JavaScript V8, який використовується в Chrome, забезпечує високу продуктивність та швидкодію

виконання коду. Node.js також відома своєю модульною архітектурою, що дозволяє розробникам використовувати власні модулі та сторонні пакети, щоб швидко розширювати можливості своїх додатків. Загалом, Node.js є потужною та ефективною платформою для створення серверних додатків, яка дозволяє створювати швидкі, масштабовані та ефективні веб-сервери.

MongoDB – це потужна, гнучка та швидка NoSQL база даних, яка забезпечує горизонтальне масштабування та можливості виконання складних операцій. Основні характеристики MongoDB включають гнучкість у моделюванні даних, підтримку JSON-подібних документів, а також можливість простого масштабування бази даних за потреби. В той час як MongoDB Atlas - це повністю керований сервіс хмарної бази даних MongoDB, який надає можливість швидко створювати та масштабувати кластери MongoDB в хмарному середовищі, без необхідності управління фізичною інфраструктурою. Переваги NoSQL баз даних, таких як MongoDB, над традиційними SQL базами даних, включають гнучкість моделювання даних, адже NoSQL бази даних дозволяють зберігати дані у вигляді невпорядкованих документів, що робить їх більш гнучкими у моделюванні різноманітних типів даних. Також NoSQL бази даних зазвичай мають простішу структуру та використовують більш простий механізм запитів, що може призводити до підвищеної швидкодії та ефективності в порівнянні з SQL базами даних у випадках, коли потрібно обробляти великі об'єми даних. Окрім цього, більшість NoSQL баз даних, включаючи MongoDB, легко масштабуються горизонтально, що означає, що можна додавати нові сервери для збільшення потужності бази даних, без перерви у роботі системи.

Використання MongoDB разом з Node.js має кілька значних переваг. По-перше, як JavaScript-орієнтовані технології, MongoDB та Node.js добре взаємодіють між собою, що спрощує розробку та обслуговування серверних додатків. По-друге, і Node.js, і MongoDB підтримують асинхронні операції, що робить їх ідеальною комбінацією для створення високопродуктивних і масштабованих серверних додатків, здатних ефективно обробляти великі обсяги запитів та даних. По-третє, документоорієнтована модель даних MongoDB

відповідає структурі JSON, що є нативним форматом для JavaScript, полегшуючи зберігання та маніпулювання даними. Ця синергія між Node.js і MongoDB дозволяє швидше створювати, розгортати та масштабувати сучасні веб-додатки. Комунікація між сервером і базою даних здійснюється за допомогою бібліотеки Mongoose, яка забезпечує зручний інтерфейс для роботи з MongoDB.

4.2 Опис серверної частини системи

Для розробки серверної частини було обрано архітектуру за шарами (package by layer). Це означає, що програмний код поділено на директорії, кожна з яких відповідає певному шару обробки даних: маршрути (routes), контролери (controllers) та моделі (models). Цей підхід дозволяє чітко відокремити різні рівні обробки запитів, забезпечуючи більш організовану структуру коду та спрощуючи його підтримку і розвиток. На рисунку 4.1 зображено структуру проекту.

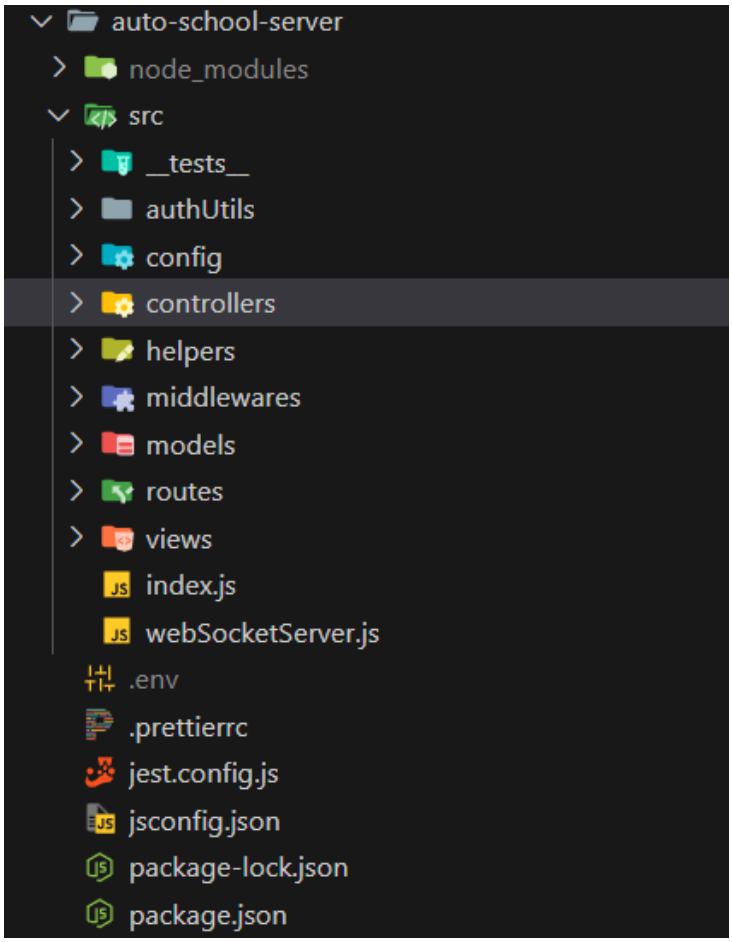


Рисунок 4.1 – Структура серверної частини системи (рисунок виконаний самостійно)

Шар маршрутів (routes) відповідає за обробку HTTP-запитів. У ньому визначаються маршрути для різних кінцевих точок (endpoint) API, які направляють запити до відповідних контролерів. Це забезпечує централізоване управління маршрутизацією та дозволяє легко додавати нові кінцеві точки або змінювати існуючі.

Для прикладу наведено декілька маршрутів для отримання, створення та видалення відгуків, використовуючи різні ролі користувачів для обмеження доступу.

```
router.get('/', reviewController.getAllReviews);

router.post(
  '/',
  authenticateJWT,
  restrictTo('student'),
  reviewController.createReview
);

router.delete(
  '/:reviewId',
  authenticateJWT,
  restrictTo('admin', 'student'),
  reviewController.deleteReview
);
```

Контролери (controllers) містять бізнес-логіку обробки запитів. Вони приймають запити від маршрутизаторів, взаємодіють з моделями для отримання або маніпулювання даними та повертають відповідь клієнту. Це дозволяє чітко розділити логіку обробки запитів і взаємодії з даними, роблячи код більш читабельним і підтримуваним.

Як приклад наведено контролер перевірки електронної пошти, який здійснює перевірку наявності користувача, терміну дії та валідності токену підтвердження.

```
exports.verifyEmail = catchAsync(async (req, res, next) => {
  const userLoginData = await userLogin.findOne({ userId:
req.params.userId });

  if (!userLoginData)
    return next(new AppError('There is no user with that id', 400));
```

```
        if (!req.query.token)
          return next(new AppError('Please, provide confirmation token',
400));

        if (new Date(userLoginData.confirmationTokenExpires) < new Date())
{
          await updateVerification(userLoginData, 'failed');
          return next(new AppError('The token has expired', 400));
        }

        if (userLoginData.confirmationToken !== req.query.token) {
          await updateVerification(userLoginData, 'failed');
          return next(new AppError('The token is invalid', 400));
        }

        await updateVerification(userLoginData, 'verified');

        res
          .status(200)
          .json({ message: 'Your email address was verified
successfully' });
      });
```

Моделі (models) визначають структуру даних та взаємодію з базою даних. Вони використовуються контролерами для виконання CRUD-операцій (створення, читання, оновлення, видалення) над даними. Використання моделей дозволяє забезпечити єдиний підхід до роботи з даними та спрощує управління складними структурами даних.

Цей приклад представляє схему моделі для автентифікації користувачів, яка містить поля електронної пошти, паролю, статусу підтвердження, токену та інших параметрів.

```
      const userLoginSchema = new mongoose.Schema({
        userId: {
          type: mongoose.Schema.Types.ObjectId,
          ref: 'userAccounts',
          unique: true,
          required: [true, 'User id is required'],
        },
        email: {
          type: String,
          required: [true, 'Email is required'],
          unique: true,
          trim: true,
          lowercase: true,
          validate: {
            validator: function (v) {
              return /^([\w-\.]+(\+\w+)?@([\w-]+\.)+[\w-]{2,4})$/.test(v);
```

```
            },
            message: (props) => `${props.value} is not a valid email
address!`,
          },
        },
        emailVerificationStatus: {
          type: String,
          enum: ['pending', 'verified', 'failed'],
          default: 'pending',
        },
        confirmationToken: String,
        confirmationTokenExpires: Date,
        passwordHash: {
          type: String,
          required: [true, 'Password is required'],
        },
        passwordChangedAt: Date,
        passwordResetToken: String,
        passwordResetExpires: Date,
        refreshToken: String,
        googleRefreshToken: String,
      });
```

Переваги обраної архітектури включають в себе чітке розділення обов’язків, адже кожен шар відповідає за окремий аспект роботи додатку, що дозволяє краще організувати код та спрощує його підтримку. Також, поділ на шари забезпечує легке масштабування додатку. Наприклад, можна окремо оптимізувати або розширювати функціонал моделей, контролерів або маршрутизаторів без впливу на інші частини системи. Окрім цього, завдяки чіткій структурі шарів, тестування окремих компонентів стає більш простим і ефективним. Можна легко створювати юніт-тести для моделей, контролерів та маршрутизаторів. Таким чином, обрана архітектура за шарами дозволяє створити добре структурований, підтримуваний та масштабований серверний додаток, що відповідає сучасним вимогам розробки програмного забезпечення.

Невід'ємною складовою структури серверів, написаних за допомогою фреймворку Express.js, є так зване middleware. Middleware в Express - це проміжне програмне забезпечення, яке виконується між отриманням запиту (request) і відправленням відповіді (response) [10]. Вони обробляють запити до їх передачі фактичним маршрутам або контролерам, виконуючи різноманітні завдання, такі як аутентифікація, авторизація, логування та обробка помилок.

Наприклад, ця функція authenticateJWT є middleware для перевірки та автентифікації користувача за допомогою JWT-токенів.

```
exports.authenticateJWT = async (req, res, next) => {
  const accessToken = req.cookies.access_token;
  if (!accessToken) {
    return res.status(401).json({ error: 'You are not logged in' });
  }
  try {
    const data = jwt.verify(accessToken,
process.env.JWT_ACCESS_TOKEN_SECRET);
    const currentUser = await UserAccount.findById(data.userId);
    if (!currentUser) {
      return res.status(400).json({ error: "User doesn't exist" });
    }
    const currentUserLoginData = await UserLogin.findOne({
      userId: data.userId,
    });
    const passwordChanged =
currentUserLoginData.passwordChangedAfter(data.iat);
    if (passwordChanged) {
      return res
        .status(401)
        .json({ error: 'Password was changed. Please, log in
again' });
    }
    req.user = currentUser;
    next();
  } catch (err) {
    return res.status(403).json({
      error: 'Access token is no longer valid',
    });
  }
};
```

Middleware дозволяє розділити логіку обробки запитів на більш малі частини, що робить код більш структурованим і підтримуваним. Вони також дозволяють пере використовувати код і застосовувати його до різних маршрутів або типів запитів.

Після реєстрації користувача надсилається лист на електронну пошту для її підтвердження за допомогою сервісу SendGrid (див. рис. 4.2). SendGrid – це хмарний сервіс для відправлення електронних листів, який забезпечує надійну доставку, масштабованість та детальну аналітику для розробників і підприємств. Після натискання на кнопку «Verify Email» користувач попадає на веб-сторінку,

яка вже відправляє відповідний запит на сервер для підтвердження електронної пошти.

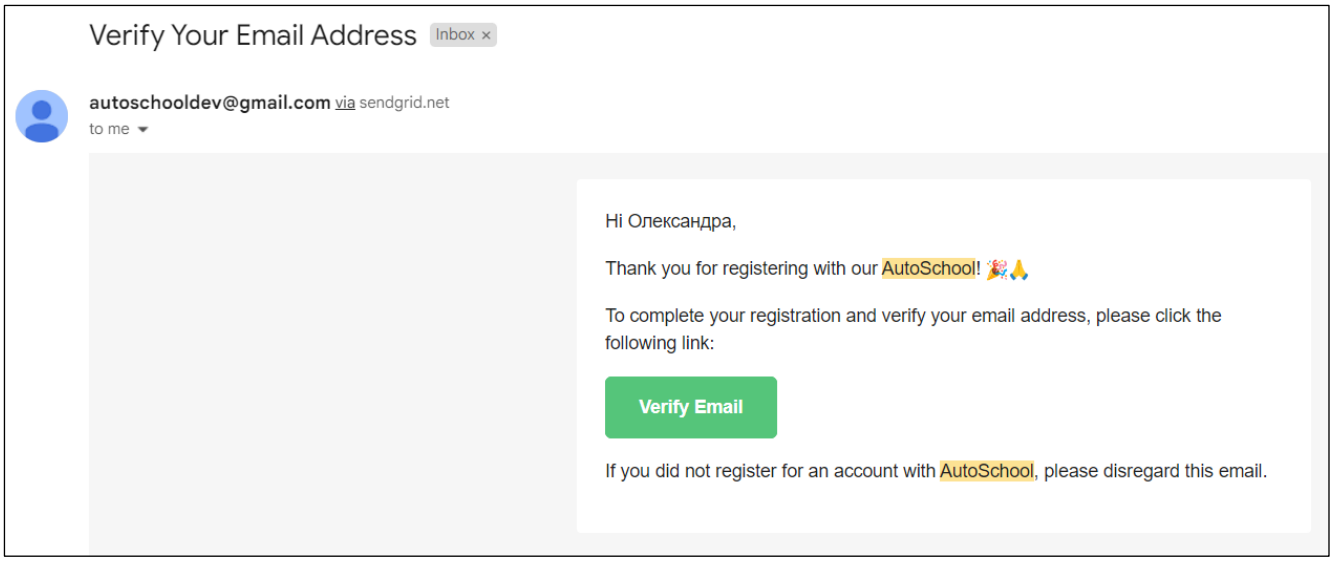


Рисунок 4.2 – Лист для підтвердження електронної пошти (рисунок виконаний самостійно)

Також в програмі є можливість завантажувати фото, яке відправляється на сервер і оброблюється за допомогою проміжного програмного забезпечення (middleware) multer, яке відповідає за обробку multipart/form-data, що в основному використовується для завантаження файлів. Тож сервер оброблює файли і зберігає у хмарному сервісі AWS S3 Bucket, який представляє собою масштабоване об'єктне сховище від Amazon Web Services для зберігання і доступу до великих обсягів даних та файлів, що забезпечує високу доступність, безпеку та інтеграцію з іншими сервісами AWS.

Нижче наведено приклад функцій, які відповідають за завантаження та стягування файлів зі сховища.

```
const {
  PutObjectCommand,
  GetObjectCommand,
  DeleteObjectCommand,
} = require('@aws-sdk/client-s3');
const { getSignedUrl } = require('@aws-sdk/s3-request-presigner');

exports.getPhotoUrl = async (s3CLient, photoName) => {
  const getPhotoObject = new GetObjectCommand({
    Bucket: process.env.BUCKET_NAME,
```

```
      Key: photoName,
    });

    const photoUrl = await getSignedUrl(s3CLient, getPhotoObject, {
      expiresIn: 3600,
    });

    return photoUrl;
  };

  exports.uploadPhotoToS3 = async (s3Client, file, fileName) => {
    const command = new PutObjectCommand({
      Bucket: process.env.BUCKET_NAME,
      Key: fileName,
      Body: file.buffer,
      ContentType: file.mimetype,
    });

    await s3Client.send(command);
  };
```

Для реалізації чату була використана бібліотека WebSocket, яка забезпечує двонаправлене, повнодуплексне зв'язування між клієнтськими і серверними додатками, що робить її ідеальною для створення чат-додатків у реальному часі. Вона дозволяє миттєво надсилати і отримувати повідомлення без необхідності постійно опитувати сервер, що зменшує затримки і підвищує ефективність. Використовуючи WebSocket, чат-додатки можуть підтримувати постійне з'єднання, забезпечуючи безперервну та швидку комунікацію між користувачами.

```
  const WebSocket = require('ws');
  const { saveMessageToDB } =
require('./controllers/chatController.js');

  const server = new WebSocket.Server({ port: 8080 });
  const users = new Map();

  server.on('connection', (ws) => {
    ws.on('message', (message) => {
      let decoder = new TextDecoder('utf-8');
      let str = decoder.decode(message);
      const parsedMessage = JSON.parse(str);

      if (parsedMessage.type === 'register') {
        const userId = parsedMessage.senderId;
        users.set(userId, ws);
        ws.userId = userId;
        ws.send(JSON.stringify({ type: 'registered' }));
        console.log(`User registered: ${userId}`);
      } else if (parsedMessage.type === 'message') {
        const recipientId = parsedMessage.recipientId;
```

```
            const text = parsedMessage.text;
            const senderId = ws.userId || parsedMessage.senderId;

            if (users.has(recipientId)) {
              const recipientSocket = users.get(recipientId);
              if (recipientSocket.readyState === WebSocket.OPEN) {
                recipientSocket.send(JSON.stringify({ senderId, recipientId,
text }));
              }
            }

            saveMessageToDB({ fromUser: senderId, toUser: recipientId,
text });
          } else if (parsedMessage.type === 'listUsers') {
            const userList = Array.from(users.keys());
            ws.send(JSON.stringify({ type: 'userList', users: userList }));
          }
        });

        ws.on('close', () => {
          users.delete(ws.userId);
          console.log(`User ${ws.userId} left chat`);
        });
      });
```

Після запису на заняття студент має можливість додати заняття як подію в свій Google календар. Для цього йому спочатку потрібно увійти до свого Google акаунту з тією поштою, яку він використав при реєстрації у системі. Для цього використовується протокол авторизації Oauth2, який дозволяє користувачам надавати веб-додаткам обмежений доступ до їхніх ресурсів на Google (наприклад, профілів, контактів, календарів) без передачі пароля. Він використовує токени доступу, щоб забезпечити безпечний та стандартизований спосіб взаємодії з API Google.

Код конфігурації Oauth2 клієнта наведений нижче.

```
const { google } = require('googleapis');

const oauth2Client = new google.auth.OAuth2(
  process.env.GOOGLE_CLIENT_ID,
  process.env.GOOGLE_CLIENT_SECRET,
  process.env.GOOGLE_REDIRECT_URL
);

const generateGoogleAuthUrl = (...scope) => {
  return oauth2Client.generateAuthUrl({
    access_type: 'offline',
    scope,
  });
```

```
};
const googleCalendar = (oauth2Client) => {
  return google.calendar({ version: 'v3', auth: oauth2Client });
};
```

Після успішного входу та надання дозволу додатку використовувати календар, користувач може додати заняття до свого календарю.

```
exports.addLessonToGoogleCalendar = catchAsync(async (req, res, next)
=> {
  const lesson = await LessonModel.findById(req.params.lessonId);
  const event = {
    summary: 'Lesson in AutoSchool',
    start: {
      dateTime: createGoogleDateTime(lesson.date, lesson.fromHour),
    },
    end: {
      dateTime: createGoogleDateTime(lesson.date, lesson.toHour),
    },
  };
  const user = await UserLoginModel.findOne({ userId:
req.user._id });
  const refreshToken = user.googleRefreshToken;
  if (!refreshToken)
    return next(
      new AppError(
        'You have to login with google to add lesson to calendar',
        400
      )
    );
  oauth2Client.setCredentials({ refresh_token: refreshToken });
  await googleCalendar(oauth2Client).events.insert({
    calendarId: 'primary',
    resource: event,
  });
  res.status(200).json({
    status: 'success',
    message: 'Event added to Google Calendar',
  });
});
```

Таким чином було розглянуто реалізований функціонал та структуру серверної частини програмної системи, яка дозволить інструкторам, студентам та адміністраторам автошколи організовувати практичні заняття в автошколі більш зручним способом.

5 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Обґрунтування вибору виду тестування

Тестування сервера є надзвичайно важливою складовою процесу розробки програмного забезпечення з кількох причин. По-перше, тестування допомагає виявити помилки та дефекти у програмному коді, які можуть призвести до непередбачуваної поведінки або витрат часу на виправлення у майбутньому. Воно сприяє покращенню якості програмного забезпечення та забезпечує більш стабільну та надійну роботу сервера. Крім того, тестування дозволяє переконатися, що сервер відповідає специфікаціям і вимогам, встановленим для нього, а також гарантує сумісність з іншими компонентами системи. Нарешті, воно забезпечує впевненість у функціональності сервера та допомагає у підтримці його працездатності під час розвитку та масштабування проекту.

Для тестування було обрано кілька методів, включаючи ручне тестування за допомогою Postman, unit-тестування та тестування навантаженням. Ручне тестування за допомогою Postman забезпечує можливість вручну взаємодіяти з сервером шляхом надсилання HTTP-запитів та перевірки відповідей. Unit-тестування дозволяє перевірити правильність роботи окремих функцій або модулів програмного забезпечення. Тестування навантаженням використовується для оцінки продуктивності та стабільності сервера під великими обсягами запитів. Комбінація цих методів допомагає забезпечити високу якість та надійність сервера.

5.2 Опис тестування

Postman – це інструмент, який використовувався для ручного тестування сервера. Він дозволяє виконувати HTTP-запити та аналізувати отримані відповіді, спрощуючи процес тестування API. Наприклад, для перевірки правильності входа в систему (login) користувача був виконаний POST-запит з відповідними даними у форматі JSON, а потім перевірена відповідь сервера, що містить статус код 200, відповідні cookie, також дані користувача (див. рис. 5.1).

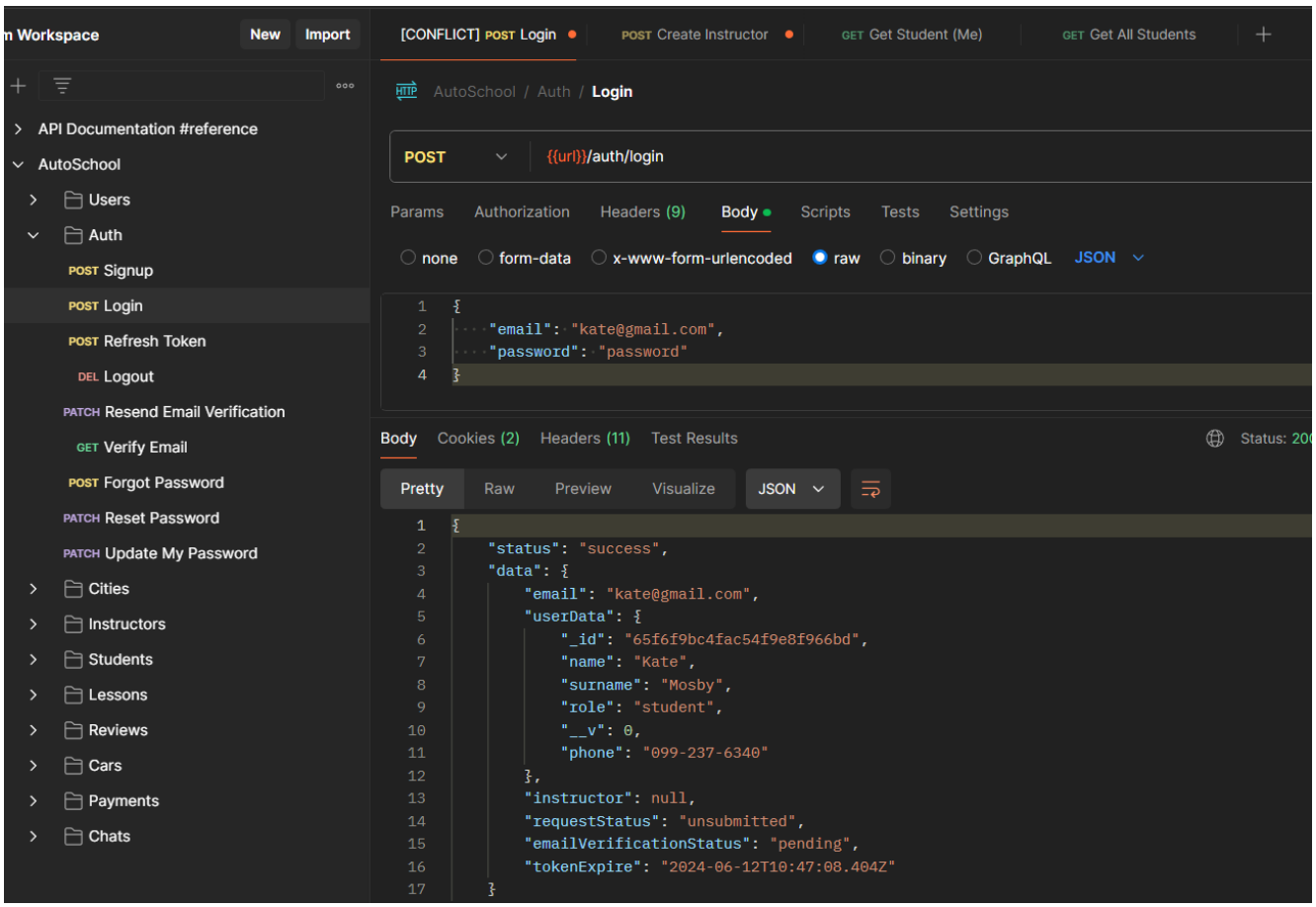


Рисунок 5.1 – Тестування за допомогою Postman (рисунок виконаний самостійно)

Postman також використовувався для тестування інших функціональних можливостей сервера, таких як аутентифікація, отримання та оновлення даних, виконання CRUD-операцій тощо. Цей інструмент дозволив легко та ефективно перевіряти функціональність сервера, забезпечуючи надійність та стабільність програмного забезпечення.

Наступним методом тестування було обрано unit-тестування. Unit-тестування – це метод тестування програмного забезпечення, який полягає в перевірці окремих компонентів (або "одиниць") коду на коректність їх роботи. Основна ідея полягає у тому, щоб тестувати кожну функцію або модуль програми окремо, в ізоляції від інших частин програми. Переваги unit-тестування включають виявлення помилок на ранніх етапах розробки, покращення структури коду, забезпечення відсутності впливу на інші частини програми, а також можливість використання тестів як документації та специфікації до коду.

Для написання unit-тестів використовувалась бібліотека Jest, яка забезпечує зручний та потужний фреймворк для тестування JavaScript-коду. Jest надає засоби для створення, виконання та аналізу тестів, а також включає в себе утиліти для створення моків та спеціальні функції для спрощення тестування асинхронного коду.

Наступний тест перевіряє, як middleware authenticateJWT поводиться у випадку недійсного доступного токену. Для ізоляції від реального процесу верифікації токену, необхідно створити mock функції jwt.verify, яка буде завжди викидати помилку "Invalid token". Таким чином, можна переконатися, що middleware коректно обробляє цю ситуацію та повертає відповідь зі статусом 403 та відповідним повідомленням про помилку.

```
describe('given not valid access token', () => {
    it('should return status 403', () => {
      req.cookies.access_token = 'invalid_access_token';

      jwt.verify = jest.fn().mockImplementationOnce(() => {
        throw new Error('Invalid token');
      });

      authenticateJWT(req, res, next);

      expect(jwt.verify).toHaveBeenCalledWith(
        req.cookies.access_token,
        process.env.JWT_ACCESS_TOKEN_SECRET
      );
      expect(res.status).toHaveBeenCalledWith(403);
      expect(res.json).toHaveBeenCalledWith({
        error: 'Access token is no longer valid',
      });
      expect(next).not.toHaveBeenCalled();
    });
  });
```

Загалом було написано 12 тестів, які покривають важливу частину працездатності програмної системи, а саме middleware та контролер, які відповідають за автентифікацію та авторизацію користувачів.

Для того, щоб дослідити стійкість системи до навантаження, було проведено тестування навантаженням у Postman. Тестування тривало 5 хвилин, з розгоном протягом перших 2 хвилин до 100 одночасних користувачів (див.

рис.5.2). За результатами тестування можна виявити, що швидкість відправки запитів на сервер досить висока, з середнім значенням 28.32 запити/сек. Це означає, що система здатна ефективно обробляти великий обсяг запитів. Середній час відповіді сервера становить 2,050 мілісекунд, що також є досить швидким. Це може свідчити про те, що система ефективно виконує запити користувачів. Загальний рівень помилок невеликий, становлячи лише 0.22% від загальної кількості запитів. Це свідчить про те, що система стабільна та має низьку ймовірність виникнення помилок.

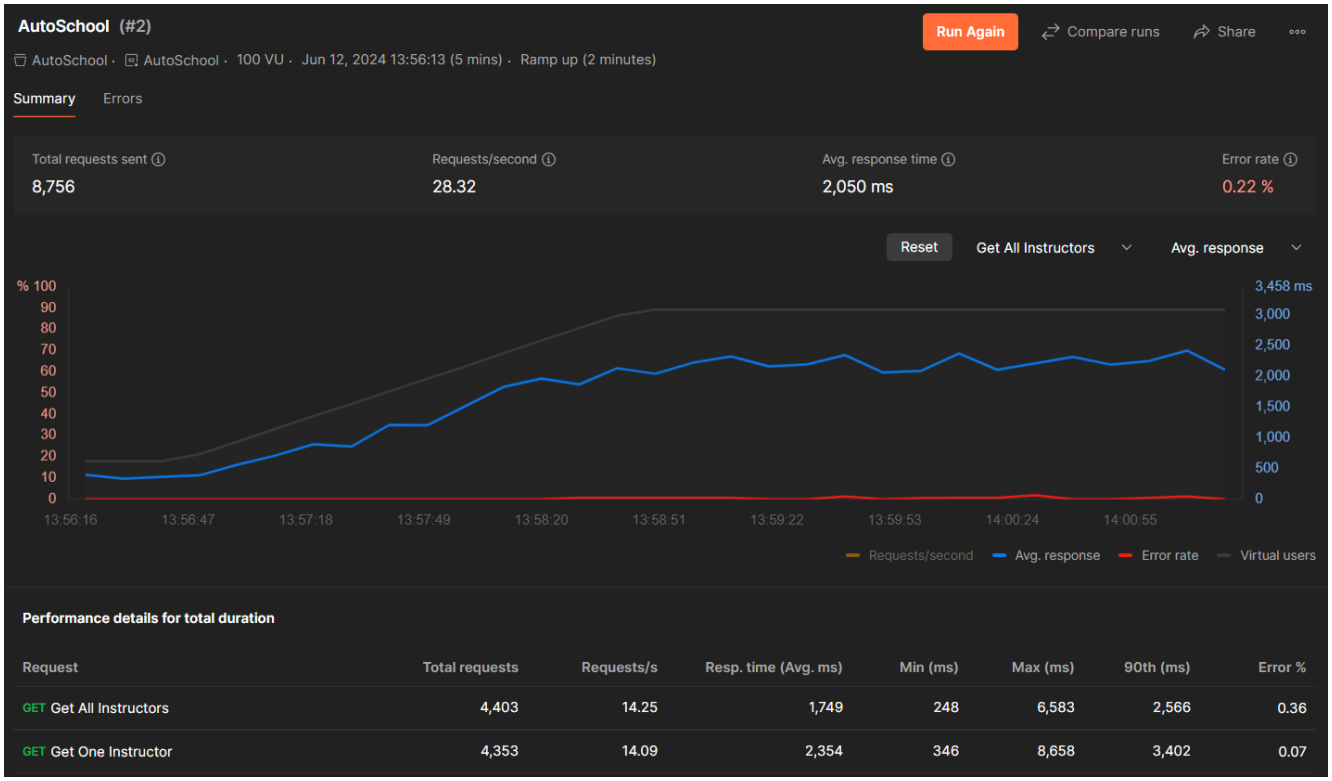


Рисунок 5.2 – Тестування навантаженням (рисунок виконаний самостійно)

Загалом, за результатами тестування навантаженням, можна зробити висновок, що система демонструє досить хорошу продуктивність та стійкість до навантаження.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було реалізовано серверну частину програмної системи для організації практичних занять в автошколі. У процесі розробки був проведений аналіз ринку та предметної галузі, що сприяло формулюванню вимог до програмної системи та описанню її архітектури. Цей аналіз дозволив визначити основний функціонал системи та виокремити її унікальні риси, які здатні привернути увагу користувачів у середовищі конкуренції.

Основним функціоналом спроектованої системи є перегляд та запис до інструкторів, перегляд їх розкладу, запис на заняття з їх оплатою, можливість подивитись та залишити відгук, спілкування з інструктором в чаті, а також відстеження власного прогресу з водіння за допомогою чек-листа необхідних вмінь.

Для розроблення серверної частини системи було обрано мову програмування JavaScript на платформі Node.js з використанням фреймворку Express. У якості бази даних було вирішено використовувати нереляційну хмарну базу даних MongoDB Atlas.

Розроблена серверна частина програмної системи для цифровізації організування практичних занять в автошколі може задовільнити усіх учасників навчального процесу в автошколі, починаючи зі студентів та інструкторів, закінчуючи адміністраторами. Її можливості сприятимуть оптимізації процесу навчання, роблячи його більш доступним і привабливим для всіх учасників.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Звіт сервісного центру МВС, 2023, https://hsc.gov.ua/wp-content/uploads/2024/03/Zvit_2023.pdf.

2. "Autopilot and Full Self-Driving Capability", дата звернення 10 травня 2024, https://www.tesla.com/support/autopilot.

3. Aashi Gupta, "Learning Node.js and MongoDB for Web Development", Search My Expert Blog, 14 лютого 2024, https://blog.searchmyexpert.com/nodejs-and-mongodb/#:~:text=One%20of%20the%20most%20significant,to%20efficiently%20handle%20JSON%20data.

4. Юлія Каграманова, "Як будувати UML-діаграми. Розбираємо три найпопулярніші варіанти", DOU, 27 жовтня 2022, https://dou.ua/forums/topic/40575/.

5. Suni L. Bangar, S. Gupta, M. Dalal та A. Inamdar, "Using Node.Js to Build High Speed and Scalable Backend Database Server", International Journal of Research in Advent Technology 4 (2016): XXXX, https://www.researchgate.net/publication/301788361_Using_NodeJs_to_Build_High_Speed_and_Scalable_Backend_Database_Server.

6. Soham Dutta, "MongoDB vs. Mongoose: Understanding the Differences and Ideal Use Cases", Data Engineering (блог), 18 грудня 2023, https://www.sprinkledata.com/blogs/mongodb-vs-mongoose-understanding-the-differences-and-use-cases#:~:text=Mongoose%20allows%20developers%20to%20define,and%20adherence%20to%20predefined%20rules.

7. Mithun Satheesh, Bruno Joseph D'mello та Jason Krol, Web Development with MongoDB and NodeJS, 2-ге вид. (Packt Publishing Ltd, 2015), XXXX.

8. Bucko, Ahmet, Kamer Vishi, Bujar Krasniqi, and Blerim Rexha. 2023. "Enhancing JWT Authentication and Authorization in Web Applications Based on User Behavior History" Computers 12, no. 4: 78. https://doi.org/10.3390/computers12040078.

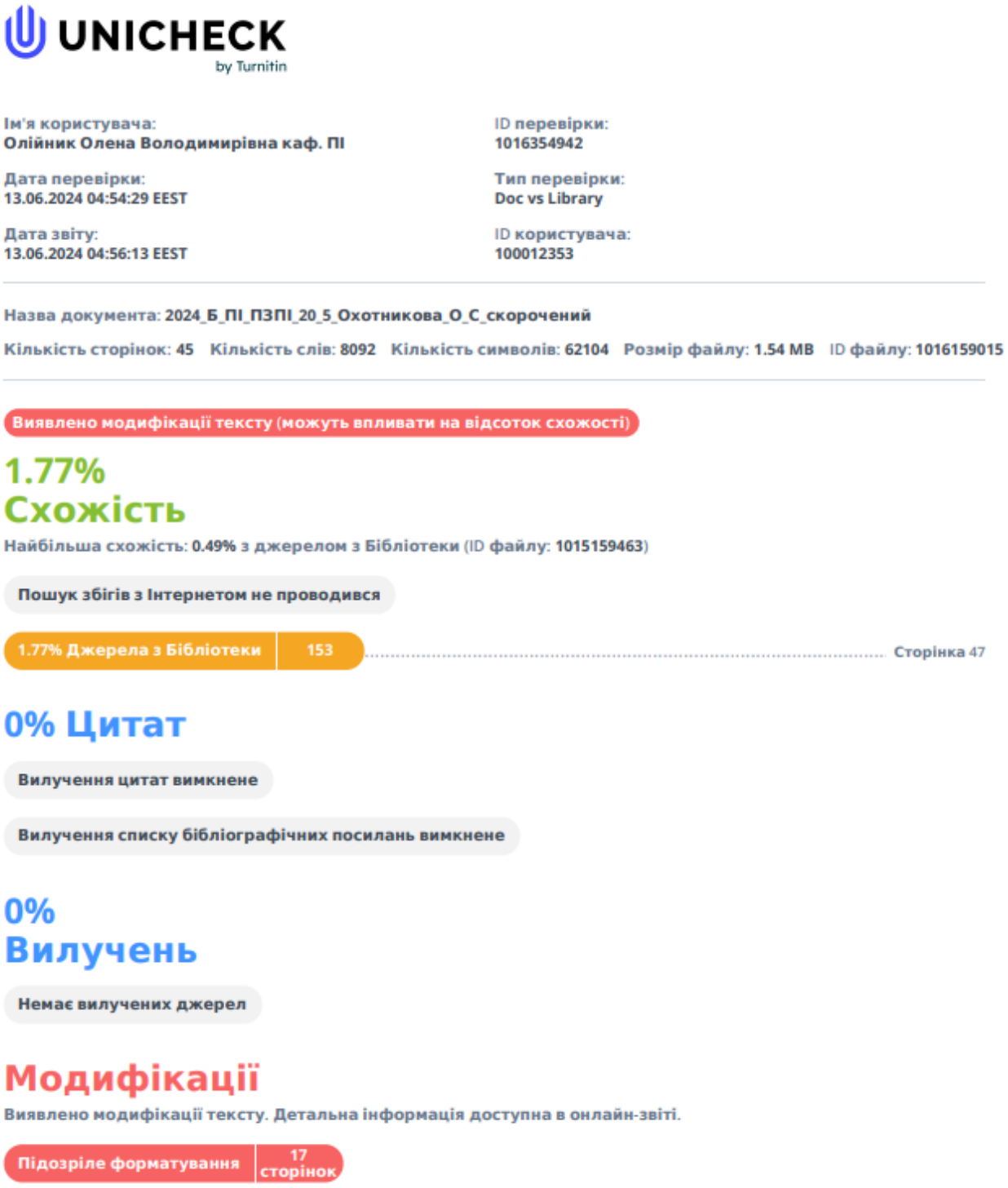
9. Олександра Сергіївна Охотникова, "Авторизація за допомогою jwt токенів", у Сучасні аспекти та перспективні напрямки розвитку науки: матеріали

VII Міжнародної студентської наукової конференції (Вінниця: UKRLOGOS Group, 2024), XXXX.

10. "Using middleware", Express, дата звернення 11 червня 2024, https://expressjs.com/en/guide/using-middleware.html.

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ



UNICHECK
by Turnitin

Ім'я користувача:
Олійник Олена Володимирівна каф. ПІ

ID перевірки:
1016354942

Дата перевірки:
13.06.2024 04:54:29 EEST

Тип перевірки:
Doc vs Library

Дата звіту:
13.06.2024 04:56:13 EEST

ID користувача:
100012353

Назва документа: 2024_Б_ПІ_ПЗПІ_20_5_Охотникова_О_С_скорочений

Кількість сторінок: 45 Кількість слів: 8092 Кількість символів: 62104 Розмір файлу: 1.54 MB ID файлу: 1016159015

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

1.77%
Схожість

Найбільша схожість: 0.49% з джерелом з Бібліотеки (ID файлу: 1015159463)

Пошук збігів з Інтернетом не проводився

1.77% Джерела з Бібліотеки 153 ..... Сторінка 47

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування 17 сторінок

ДОДАТОК Б

Слайди презентації

Харківський національний університет радіоелектроники

Програмна система для організації практичних занять в автошколі. Back-end

Виконала: ст. гр. ПЗПІ-20-5 Охотникова О. С.
Керівник: ст. викл. кафедри ПІ Олійник О. В.

Захист кваліфікаційної роботи | 2024

2

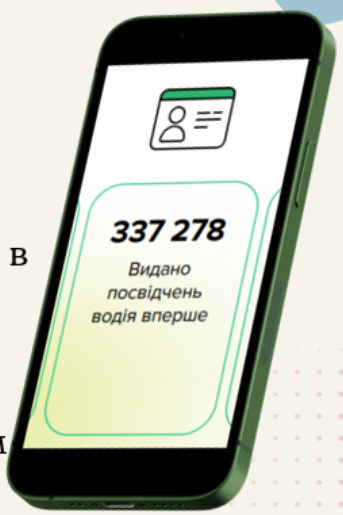
МЕТА РОБОТИ

Створення серверної частини комплексної програмної системи для організації практичних занять в автошколі, яка надасть зручний інтерфейс для запису на заняття, відстеження прогресу та загалом підвищить якість навчального процесу під час проходження практичного курсу з водіння.

3

АКТУАЛЬНІСТЬ

- Свобода пересування та незалежність від громадського транспорту
- Для складання практичного іспиту навчання в автошколі є обо'язковим
- За статистикою МВС України в 2023 році посвідчення водія було видане 337 278 особам

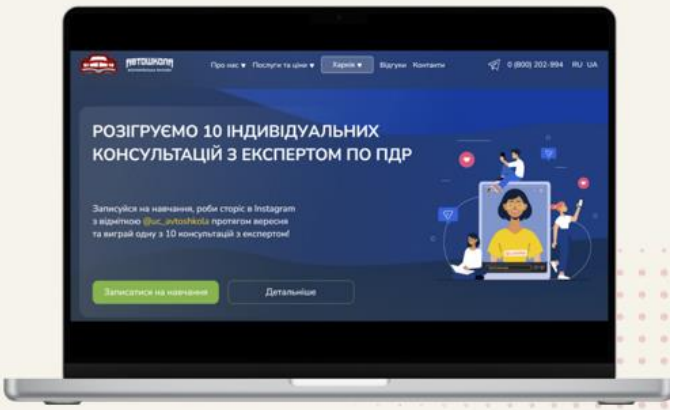


4

АНАЛІЗ КОНКУРЕНТІВ

"Всекураїнська автошкола онлайн"

+ Telegram боти для консультації
+ Перелік інструкторів та машин за містом
+ Можна записатись на заняття
- Не можна залишити відгук
- Не має відгуків про інструкторів
- Не можна сплатити заняття
- Не можна записатись до інструктора з іншого міста

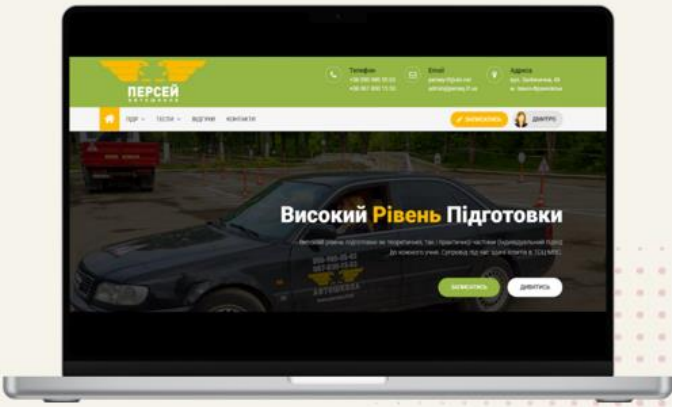


5

АНАЛІЗ КОНКУРЕНТІВ

Автошкола “Персей”

+ Результати тестів зберігаються
+ Велика база теорії та тестів
- Неконтрольоване залишання відгуків
- Не має інформації про інструкторів
- Не можна записатись на заняття
- Не має панелі адміністратора



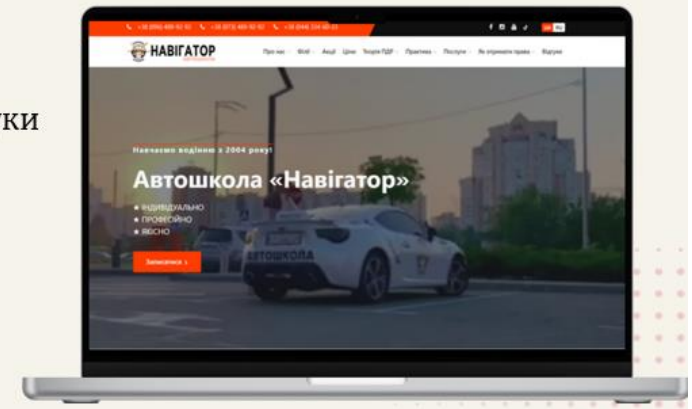
6

АНАЛІЗ КОНКУРЕНТІВ

Автошкола “Навігатор”

+ Є інформація про інструкторів
+ Можна залишити та подивитись відгуки
+ Авторизація з Google або Facebook
+ Онлайн-чат для консультації

- Не можна записатись на заняття
- Не можна сплатити заняття
- Веб-сайт довго завантажується



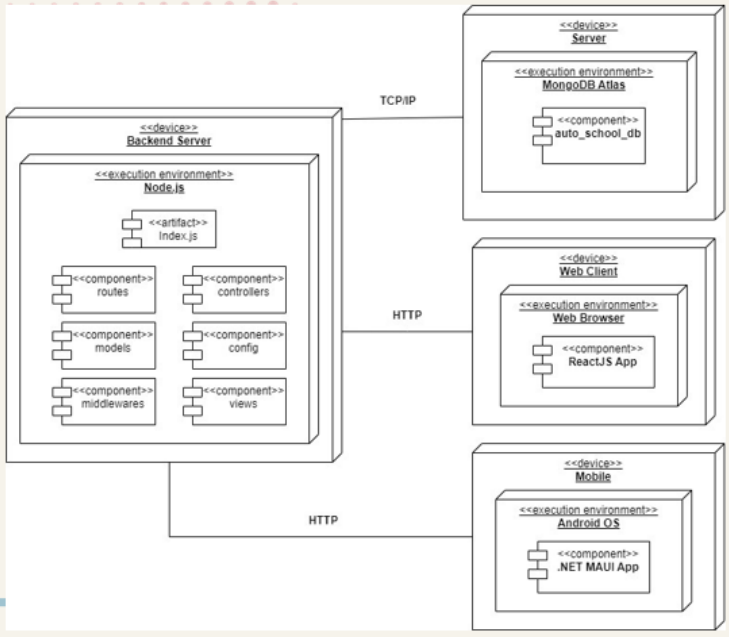
7

ВИСНОВКИ АНАЛІЗУ КОНКУРЕНТІВ

- Більшість конкурентів надають лише інформаційні послуги.
- Функціонал обмежений формою для заявки.
- Деякі конкуренти фокусуються на теоретичній частині (вивчення матеріалу, тести).
- Мало хто пропонує корисний функціонал для практичних занять і їх оплати.
- Використання сторонніх платформ знижує контроль і збільшує ймовірність помилок.

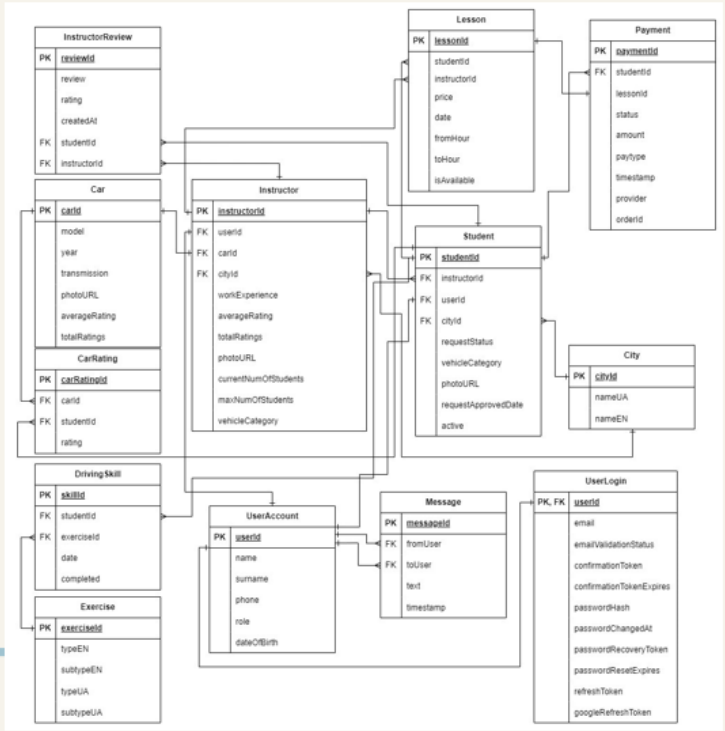
8

АРХІТЕКТУРА СИСТЕМИ



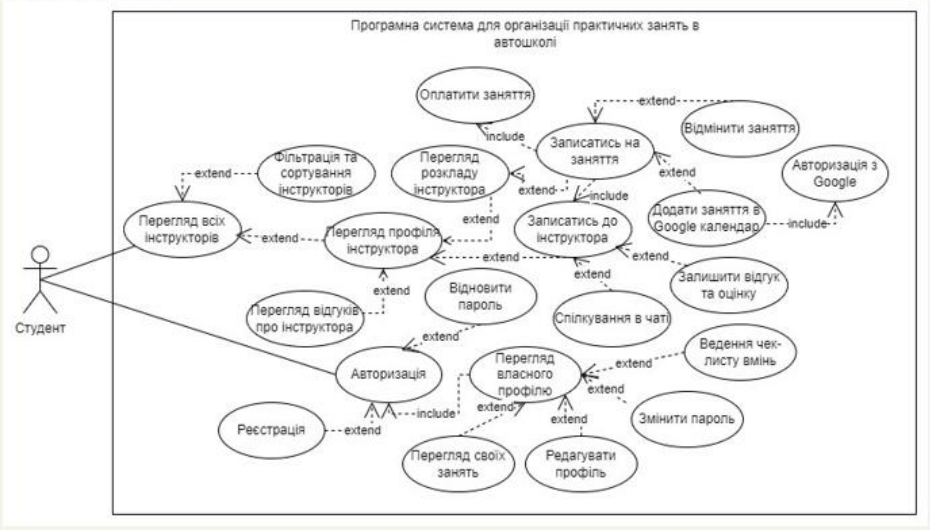
9

ПРОЕКТУВАННЯ БАЗИ ДАНИХ



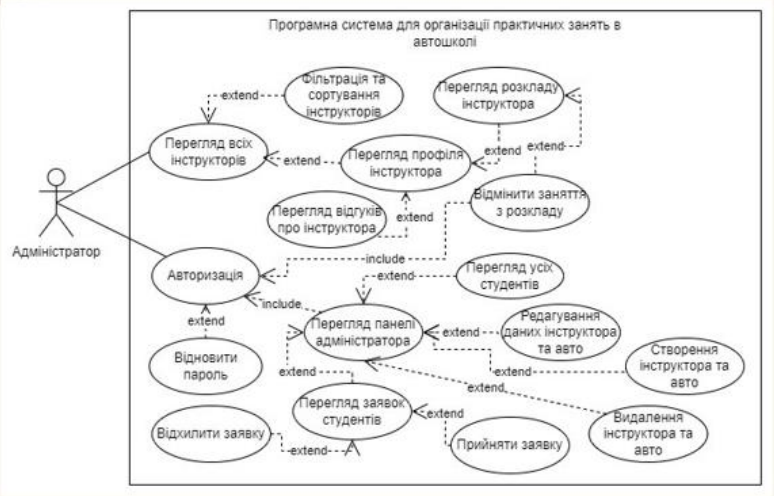
10

СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ



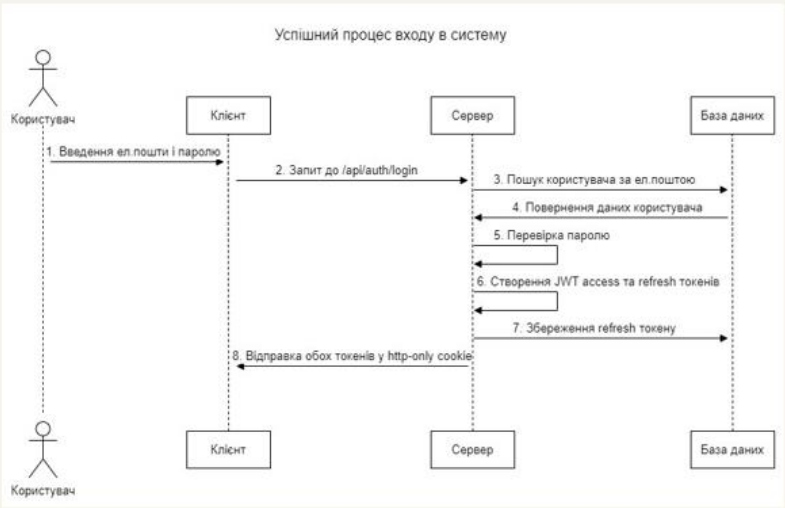
СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ

11



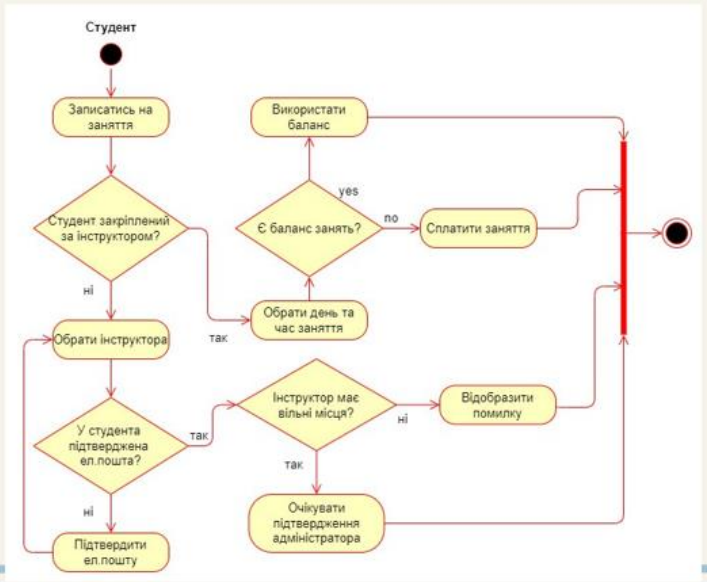
ДІАГРАМА ВХОДУ В СИСТЕМУ

12



13

БІЗНЕС ЛОГІКА



14

ОБРАНІ ТЕХНОЛОГІЇ

JS

node

express

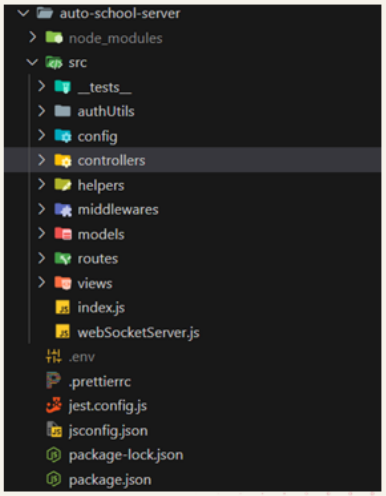
mongoDB

15

СТРУКТУРА СЕРВЕРУ

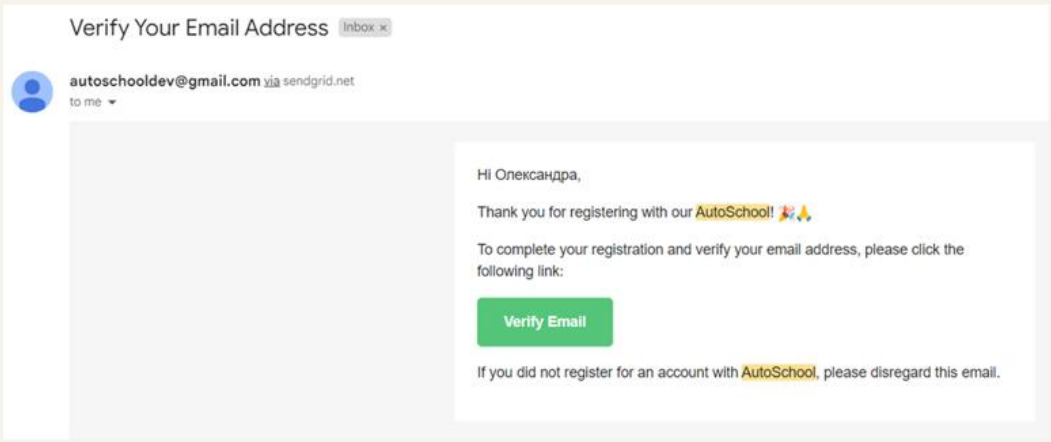
Маршрут
и
Контролер
и
Моделі

Middleware



16

ПІДТВЕРДЖЕННЯ ЕЛ. ПОШТИ



17

ПРИКЛАД КОДУ

```
exports.getAllInstructors = catchAsync(async (req, res, next) => {
  let instructorsQuery = new APIFeatures(InstructorModel.find(), req.query)
    .filter()
    .sort()
    .paginate();

  const instructors = await instructorsQuery.query
    .populate('car')
    .populate('city')
    .exec();

  for (const instructor of instructors) {
    instructor.photoURL = await getPhotoUrl(s3, instructor.photoURL);
    instructor.car.photoURL = await getPhotoUrl(s3, instructor.car.photoURL);
  }

  res.status(200).json({
    status: 'success',
    results: instructors.length,
    data: instructors,
  });
});
```

Контролер для отримання всіх інструкторів

18

ПРИКЛАД КОДУ

```
exports.requestAssignInstructor = catchAsync(async (req, res, next) => {
  // Check if student verified email
  const studentLoginData = await UserLoginModel.findOne({
    userId: req.user._id,
  }).select('emailVerificationStatus');

  if (studentLoginData.emailVerificationStatus !== 'verified')
    return next(new AppError('You have to verify email address first', 403));

  // Check if instructor is available
  const instructor = await InstructorModel.findById(req.params.instructorId);

  if (!instructor || !instructor.available)
    return next(
      new AppError('There is no available instructor with that id', 400)
    );

  await StudentModel.findOneAndUpdate(
    { userId: req.user._id },
    { instructorId: req.params.instructorId, requestStatus: 'pending' }
  );

  res.status(200).json({
    status: 'success',
    message:
      'Your request is being processed. Please, wait for the administrator to approve it',
  });
});
```

Контролер для обробки запиту на закріплення за інструктором

19

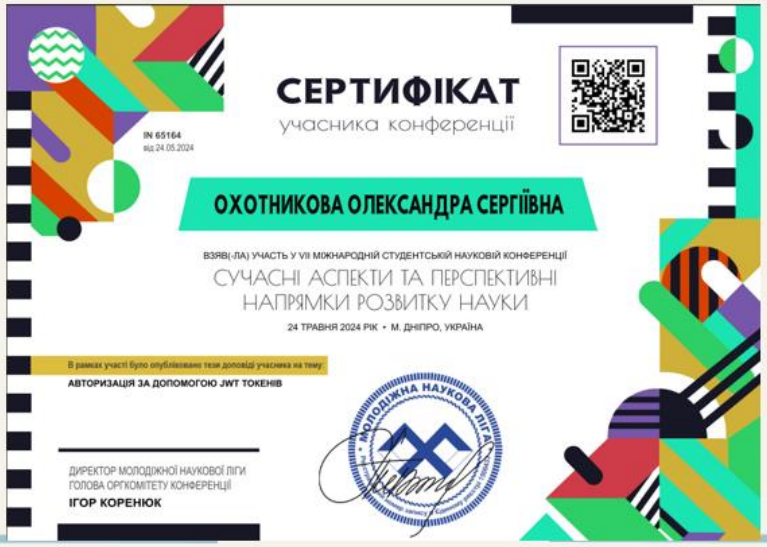
ТЕСТУВАННЯ

- Тестування у Postman
- Unit-тестування
- Load-тестування

```
describe('given not valid access token', () => {
  it('should return status 403', () => {
    req.cookies.access_token = 'invalid_access_token';

    jwt.verify = jest.fn().mockImplementationOnce(() => {
      throw new Error('Invalid token');
    });

    authenticateJWT(req, res, next);

    expect(jwt.verify).toHaveBeenCalledWith(
      req.cookies.access_token,
      process.env.JWT_ACCESS_TOKEN_SECRET
    );
    expect(res.status).toHaveBeenCalledWith(403);
    expect(res.json).toHaveBeenCalledWith({
      error: 'Access token is no longer valid',
    });
    expect(next).not.toHaveBeenCalled();
  });
});
```

20

АПРОБАЦІЯ РЕЗУЛЬТАТІВ



21

ВИСНОВКИ

- Була реалізована і протестована серверна частина програмної системи для організації практичних занять в автошколі.
- Проведений аналіз ринку та предметної області, сформовані вимоги до системи.
- Основний функціонал включає перегляд інструкторів, запис на заняття, оплату, відгуки, чат з інструктором та відстеження прогресу.
- Для реалізації використано: JavaScript, Node.js, Express.js, MongoDB Atlas.
- Система задовольняє потреби студентів, інструкторів та адміністраторів, оптимізуючи процес навчання та роблячи його більш доступним і привабливим.

ДОДАТОК В

Тези доповіді

СЕКЦІЯ 16.

КОМП'ЮТЕРНА ТА ПРОГРАМНА ІНЖЕНЕРІЯ

Охотникова Олександра Сергіївна, здобувачка вищої освіти
факультету комп'ютерних наук
Харківський національний університет радіоелектроніки, Україна

Науковий керівник: Олійник Олена Володимирівна, старший викладач
кафедри програмної інженерії
Харківський національний університет радіоелектроніки, Україна

АВТОРИЗАЦІЯ ЗА ДОПОМОГОЮ JWT ТОКЕНІВ

У сучасному світі веб-розробки питання безпеки та авторизації користувачів набувають все більшого значення. Розробникам необхідно використовувати надійні інструменти для забезпечення безпеки і контролю доступу до даних, одним з яких є веб-токени JSON (JWT). Наразі JWT токени є одним із найпопулярніших засобів авторизації.

Веб-токен JSON (JWT) — це відкритий стандарт (RFC 7519), який визначає компактний і самодостатній спосіб безпечної передачі інформації між сторонами в якості JSON об'єкту. Цю інформацію можна перевірити та довіряти їй, оскільки вона має цифровий підпис. JWT можна підписати за допомогою секрету (з алгоритмом HMAC) або пари відкритих/приватних ключів за допомогою RSA або ECDSA [1]. Тобто у своїй основі JWT – це спосіб перевірки автентичності даних, які записані у JSON форматі, адже цифровий підпис гарантує що вміст даних не було підроблено під час зберігання або передачі [2]. Варто також підкреслити, що JWT забезпечує підтвердження права власності на дані, але не їх шифрування. Це означає, що будь-хто, хто перехопить токен, зможе побачити дані токену декодувавши його [3].

Веб-токени JSON складаються з трьох частин, розділених крапками (.), які є закодованими рядками у base64url: заголовок (header), дані (payload) та підпис (signature) [1]. Це означає, що кожна з цих частин є JSON об'єктом, який потім кодується за допомогою base64url для утворення 3 частин JWT. Перша частина, заголовок, зазвичай складається з типу токену, що в цьому випадку є JWT та використовуваний алгоритм підпису, наприклад HMAC SHA256 або RSA. Наступна частина містить дані або претензії (claims) про сутність, такі як id користувача в системі, а також додаткові дані, наприклад, iss (дата видачі токена), exp (дата його закінчення) та інші. На останок, щоб створити останню частину, потрібно взяти закодовані перші дві частини, секрет (рядок, який розробник повинен зберігати в безпечному місці, як-от .env файл), алгоритм, указаний у заголовку, і підписати це секретом [1].

В основному є 2 способи використання JWT токенів: авторизація та передача даних. Однак, у цьому документі ми зосередимося виключно на принципах авторизації. Отож, найпоширенішим сценарієм використання JWT є авторизація. Після першого входу користувача у додаток кожен наступний запит включатиме

токен, що дозволить користувачеві отримати доступ до послуг і ресурсів, які дозволені цьому токену.

Розглянемо більш детально принципи авторизації користувачів за допомогою JWT-токенів. Коли користувач входить до системи, після успішної автентифікації сервер генерує JWT токени та відправляє їх клієнту. Рекомендується, щоб весь зв'язок між клієнтом і сервером використовував протокол HTTPS, щоб запобігти підслуховуванню мережі та атакам типу «людина посередині» (man-in-the-middle). У свою чергу, клієнт зберігає токени у локальному сховищі або файлах cookie. Утім, рекомендується зберігати обидва токени у HTTP-only cookie. Це означає, що вони будуть доступні лише для серверних запитів і недоступні для JavaScript, що знижує ризик зловживання атаками міжсайтового скриптування (XSS) [4].

Після отримання токенів, для кожного наступного запиту, який вимагає авторизованого користувача, клієнт надсилає JWT у заголовках запиту або прикріпляє HTTP-only cookie за допомогою властивості setCredentials. Сервер перевіряє токен, перевіряючи підпис і декодуючи дані, щоб переконатися в автентичності та авторизації користувача.

Зокрема, в цій схемі авторизації використовуються два типи токенів: короткочасний access токен та довготривалий refresh токен. Зберігаючи довготривалий refresh токен у базі даних, система дозволяє користувачам оновлювати свій access токен без необхідності повторної авторизації кожного разу, коли він закінчується. Це забезпечує довготривалу сесію користувача та зручність використання системи. Короткотривалість access токену необхідна з міркувань безпеки, адже якщо токен буде перехоплений зловмисником, то він зможе здійснити вхід до системи від імені користувача.

У разі компрометації refresh токену, зловмисник буде мати доступ до системи до закінчення терміну дії токену. Щоб зменшити цей ризик, можна застосувати функцію ротації токенів. Ротація токенів — це процес періодичної видачі нових токенів на заміну існуючим Є кілька способів здійснення цього, наприклад, видача нової пари токенів на основі певних подій, таких як зміна пароля користувача [5].

На завершення, впровадження JSON Web Tokens (JWT) в веб-застосунках значно підвищує рівень безпеки та ефективності автентифікації і обміну даними. Завдяки своїй простоті, безстанності (stateless) та надійності, JWT стають незамінним інструментом для сучасних веб-розробників. Використання передових практик і глибоке розуміння принципів роботи з JWT дозволяють створювати безпечні, масштабовані та надійні системи, які відповідають вимогам сьогодення і забезпечують захист даних користувачів.

Список використаних джерел:

1. Introduction to JSON Web Tokens. URL: https://jwt.io/introduction (дата звернення: 18.05.2024).
2. Copes F. JWT authentication: Best practices and when to use it. URL: https://blog.logrocket.com/jwt-authentication-best-practices/#:~:text=The%20server%20sends%20the%20JWT,need%20for%20constant%20database%20lookups. (дата звернення: 18.05.2024).
3. JWT Debugger. URL: https://jwt.io/#debugger-io (дата звернення: 18.05.2024).
4. Krishna V. Using JWTs as HttpOnly cookies with React.js. URL: https://vivekkrishnavk.medium.com/using-jwts-as-http-only-cookies-with-react-js-a301991fdfa6 (дата звернення: 18.05.2024).
5. Shaw A. JWT Token Rotation and Reuse Detection. URL: https://aryanshaw.hashnode.dev/jwt-token-rotation-and-reuse-detection (дата звернення: 18.05.2024).

СЕРТИФІКАТ

учасника конференції

IN 65164
від 24.05.2024

ОХОТНИКОВА ОЛЕКСАНДРА СЕРГІЇВНА

ВЗЯВ(-ЛА) УЧАСТЬ У VII МІЖНАРОДНІЙ СТУДЕНТСЬКІЙ НАУКОВІЙ КОНФЕРЕНЦІЇ

СУЧАСНІ АСПЕКТИ ТА ПЕРСПЕКТИВНІ НАПРЯМКИ РОЗВИТКУ НАУКИ

24 ТРАВНЯ 2024 РІК • М. ДНІПРО, УКРАЇНА

В рамках участі було опубліковано тези доповіді учасника на тему:

АВТОРИЗАЦІЯ ЗА ДОПОМОГОЮ JWT ТОКЕНІВ

ДИРЕКТОР МОЛОДІЖНОЇ НАУКОВОЇ ЛІГИ
ГОЛОВА ОРГКОМІТЕТУ КОНФЕРЕНЦІЇ
ІГОР КОРЕНЮК

МОЛОДІЖНА НАУКОВА ЛІГА